NO. 41 $2.50 OCTOBER 1981

MICRO

THE 6502/6809 JOURNAL

Special PET/VIC Feature

A Look at the TRS-80 Color Computer

Step Up to Programmable Motion

A to D Converters for Your Computer

Turn your Apple into the world's most versatile personal computer.

The SoftCard™ Solution. SoftCard turns your Apple into two computers. A Z-80 and a 6502. By adding a Z-80 microprocessor and CP/M to your Apple, SoftCard turns your Apple into a CP/M based machine. That means you can access the single largest body of microcomputer software in existence. Two computers in one. And, the advantages of both.

Plug and go. The SoftCard system starts with a Z-80 based circuit card. Just plug it into any slot (except 0) of your Apple. No modifications required. SoftCard supports most of your Apple peripherals, and, in 6502-mode, your Apple is still your Apple.

CP/M for your Apple. You get CP/M on disk with the SoftCard package. It's a powerful and simple-to-use operating system. It supports more software than any other microcomputer operating system. And that's the key to the versatility of the SoftCard/Apple.

BASIC included. A powerful tool, BASIC-80 is included in the SoftCard package. Running under CP/M, ANSI Standard BASIC-80 is the most powerful microcomputer BASIC available. It includes extensive disk I/O statements, error trapping, integer variables, 16-digit precision, extensive EDIT commands and string functions, high and low-res Apple graphics, PRINT USING, CHAIN and COMMON, plus many additional commands. And, it's a BASIC you can compile with Microsoft's BASIC Compiler.

More languages. With SoftCard and CP/M, you can add Microsoft's ANSI Standard COBOL, and FORTRAN, or Basic Compiler and Assembly Language Development System. All, more powerful tools for your Apple.

Seeing is believing. See the SoftCard in operation at your Microsoft or Apple dealer. We think you'll agree that the SoftCard turns your Apple into the world's most versatile personal computer.

Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us.

SoftCard is a trademark of Microsoft. Apple II and Apple II Plus are registered trademarks of Apple Computer. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research, Inc.

MICROSOFT
CONSUMER PRODUCTS

Microsoft Consumer Products, 400 108th Ave. N.E., Bellevue, WA 98004. (206) 454-1315

The A2-GE1 Graphics Editor for the Apple II

Convenient graphics power...

The A2-GE1 Graphics Editor is a collection of programs designed to put the power of A2-3D1 and A2-3D2 graphics in your hands.

The A2-GE1 includes **Object Editor**, **Motion Programmer**, **Motion Playback**, **Slideshow Playback**, and a special A2-3D2 interface for BASIC programmers.

With **Object Editor** you can create whatever objects you want in the colors of your choice. You can also type in whatever 3D text you want, and in different sizes. And saving an object is as easy as naming it.

Then give the object names to **Motion Programmer** and see how the beautifully laid out keyboard controls will let you switch objects on or off, animate them, or add upper or lower case 2D text mixed right in.

You can also record your entire presentation, animation and all, for later use with **Motion Playback**, or just take "computer snapshots" of scenes with **Slide Show Playback**.

We've reached our goal of giving you the most sophisticated graphics utilities in the market-place . . .

See them now at your dealer!

A2-GE1 Graphics Editor
$34.95 on disk (48K and A2-3D2 required).

A2-3D1 with 3D2 Enhancement*
$84.90 on disk (48K required)

*3D1 owners may update to 3D2 for $24.95. Contact SubLOGIC for details.

subLOGIC
Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
Telex: 206995

For direct order, include $3 for UPS or $5 for first class mail delivery. Illinois residents add 5% sales tax. Visa and MasterCard accepted

MICRO™

THE 6502/6809 JOURNAL

STAFF

Editor/Publisher
ROBERT M. TRIPP

Associate Publisher
RICHARD RETTIG

Associate Editors
MARY ANN CURTIS
FORD CAVALLARI

Special Projects Editor
MARJORIE MORSE

Production Coordinator
PAULA M. KRAMER

Typesetting
EMMALYN H. BENTLEY

Advertising Manager
CATHI BLAND

Circulation Manager
CAROL A. STARK

Dealer Orders
LINDA HENSDILL

MICRO Specialists
APPLE: FORD CAVALLARI
PET: LOREN WRIGHT
OSI: PAUL GEFFEN

Comptroller
DONNA M. TRIPP

Bookkeeper
KAY COLLINS

Sales Representative
KEVIN B. RUSHALKO
603/547-2970

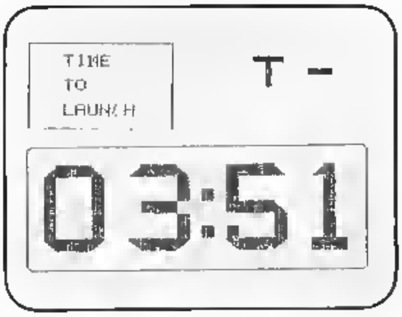
ARTICLES

PET BONUS

APPLE BONUS

DEPARTMENTS

About the Cover



Our cover picture this month, taken at Kennedy Space Center, highlights the intensified space activity taking place this year. In April, the Space Shuttle Columbia successfully completed its historic first mission. In August, Voyager II completed its close encounter with Saturn. And this month, Columbia is getting its second run, marking the first re-use of a spacecraft.

The manufacturer of a popular data base manager for the PET has announced that its product, along with the PET, is being used by NASA to keep various logs for its ground-based vehicles. An Apple computer is scheduled to monitor experiments in an upcoming mission of the Shuttle-based Space Lab. As our knowledge of our cosmos and our world increases, we're sure that the role of the microprocessor will continue to increase, too.

MICRO is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824 and Avon, MA 02322
USPS Publication Number: 483470
ISSN: 0271-9002

Send subscriptions, change of address, USPS Form 3579, requests for back issues and all other fulfillment questions to
MICRO
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515

Subscription rates	*Per Year*
U.S.	$18.00
Foreign surface mail	$21.00
Air mail:	
Europe	$36.00
Mexico, Central America	$39.00
Middle East, North Africa	$42.00
South America, Central Africa	$51.00
South Africa, Far East, Australasia	$60.00

MICRO Editorial

The analysis of the reader questionnaire has given the MICRO staff a good idea of the microcomputer systems our readers use. As mentioned last month, we were surprised at the large number of readers who have access to more than one 6502- or 6809-based system. One reason that this is an important statistic is that it underscores MICRO's role as a general 6502/6809 resource journal. While MICRO has presented, and will continue to present, quality system-specific articles and software, our main charge is to examine the similarities between the various systems we cover. These similarities manifest themselves at almost all levels, from the processor up to the high-level languages supported. For example, Applesoft, OSI, PET, and TRS-80 Color BASIC are all members of the popular Microsoft family of BASICs. And several other languages, including Pascal and FORTH, are generally compatible from system to system. Thus, most of the high level applications we receive, all of the discussions on programming technique, and most tutorials on programming languages, tend to be of as much universal interest as our articles on the 6502 and 6809 processors. Even some system-specific articles we publish can be generalized due to these compatibilities between systems.

Articles useful to more than one system are more valuable. Even though the bulk of articles received are written with one system in mind, that does not mean their usefulness stops with that one system. Most ideas, with the exception of some fancy I/O or screen-oriented tricks, can be transported from one system to the next. To implement this generalization you need access to several systems and knowledge of their basic characteristics.

Right now MICRO's lab has the hardware to generalize. We have an Apple, several PETs, an Atari, a TRS-80 Color, KIM, SYM, AIM, and an OSI, and soon they will all be communicating through a Flexi-Plus I/O board. Eventually, we'll be able to easily transfer files between systems. However, what the MICRO staff lacks is the time to generalize all but a few of the articles we receive. So we put it to the readers, especially those with two or more systems.... Are any of you interested in forming a "generalizing pool" which would take useful, system-specific articles, and make the minor modifications that may be necessary for implementation on other systems?

In keeping with our policy of highlighting the similarities between the systems we cover, and recognizing the benefits to the readers of tighter organization, MICRO will be separating its editorial content into several, rotating categories starting in November. In the past we've been tempted to separate the articles by system, mainly because of the simplicity of that organization. However, such a categorization totally overlooks the Apple graphics program which may be useful to the Atari user, or the OSI BASIC program which will help the PET owner. Thus, starting next month, we will be placing articles into logical, subject-oriented groups which will be rotated from month to month in coordination with our feature sections. We think you'll find that the new organization will make MICRO more attractive and more readable than it has been in the past.

Each month, MICRO receives articles in a variety of lengths. This brings up several points on the presentation of articles in MICRO. Oftentimes, we will receive a short article describing some useful concept which does not really require a long write-up. And I'm sure for every one of these short articles we receive, there exist piles of user notes which are never published. We at MICRO want everyone to know that short articles and notes are not discouraged; in fact, they may provide information unavailable in any other form. Consequently, starting in December, we will be including a short subject section in MICRO. Material appropriate for this section may include short utilities, program modifications, or any good idea that is not well-suited to article format. The material appearing in this section will not receive payment, but authors will receive full acknowledgement.

MICRO feels that this section will fill an important gap. It will provide an opportunity for the computer enthusiast who does not particularly like to write to transmit ideas to the MICRO readership. It will permit short subject matter to appear quickly, without having to wait in our backlog of articles. Finally, it will act as a forum for our readers. So if you have any interesting material which is not suited for an article but important nonetheless, please send it in for this new section.

Ford Cavallari

MICRO Letterbox

Copyright Controversy

Dear Editor:

Today I actually lectured an innocent salesman at a local computer store on the evils of copy protected software. "Lectured" might be a bit mild — I yelled at him. But, so that we may all understand better how such things happen, let me relate a bit of history.

I'm an old hand at personal computing, having built and owned one of the first Altair 8800's. Since then, I've built, owned and used a Sol 20. I built my computers to save money, not because I get any special enjoyment out of soldering and debugging hardware. I am the same way about most software. Unless I wrote it, I want to use it as simply and easily as possible. I really don't want to help the author debug it and I want it to work when I need it, which, for the business software I buy, is all of the time.

Now, I know that there are folks out there who claim that anything can be copied. In fact, a youth at a local computer store mentioned such to me. But the fact is, I'm not interested in playing cryptographer. Sure, I'm amused by the people who get their jollies out of breaking a copy protected piece of software. But, I'm a business man who bought my computer to help me in serious business ventures. I don't have time to attempt to break copy protection codes or to hunt up those people.

So, I feel that when I lay down a lot of cash for a piece of software, I have a right to some security in the form of copiability (for backups) or at least two backup copies with each software package I purchase. Those backups should be delivered with the software when purchased, not held for ransom until the registration card is sent in.

My new motto is "If It's Copy Protected and No Backups Come With It, Don't Buy It." The software producers must realize that they are not selling ROMs which are unlikely to be damaged. Instead, they are selling very fragile magnetic media. With a little bit of experience, no sensible business person is going to put up with this copy protection racket for long. Unless the software houses do something and do it soon on their own, they can be sure that government will be there momentarily to regulate their activities. The honest purchaser of software deserves a little consideration. Generally, he signs away his life when he fills out one of those little registration cards. The software producers have a responsibility to let us know what's in those tantalizing but uninformative packages.

Barry Gerber
Decision Points, Inc.
5339 Ventura Canyon Ave.
Van Nuys, CA 91401

Dear Editor:

In the June issue of MICRO, on page 6 in the Letterbox section, the reader states: "I freely admit there are many copyrighted programs in my library which I obtained through software swaps and from friends.... If I were using any of these for commercial gain or were reselling them *through any means*, I should be locked up." (Italics added.)

But the writer was reselling them! To sell is "to give up (property) to another for money or other valuable consideration for a price" (Webster's *7th Collegiate Dictionary*). The selling price was other programs.

However, on my first software purchase (a $100 program from one of the "name" software houses) I learned three valuable lessons:

1. Never buy a program that you cannot copy.
2. Never buy a program that you cannot modify.
3. Never buy a program until you have tried it out.

Lowell Ray Anderson
Box 67
Cody, WY 82414

Apple Bulletin Board

Dear Editor:

I would like to inform your readers of a free bulletin board for Apple users in Jacksonville, Florida. The "SEB Bulletin Board" is maintained from 6 p.m. - 8 a.m., seven days a week. The access number is 904-743-7050. The system is an Apple with 48K, one Disk Drive and a DC Hayes Micromodem.

Sam Batch
SEB Computer
1705 University Blvd. North
Jacksonville, FL 32211

If you have a tip to share with our readers, a question for our editors or just an opinion you'd like to express, write to Letterbox.

We receive many letters and cannot answer each individually, but we try to publish some each month. When you write, please cover only one topic per letter and try to be brief. Also, address the letter to "Letterbox" on the envelope, and chances are, you'll be reading your own letter in MICRO.

MICRO

The Radio Shack Color Computer:

A 6809-Based System.

The Radio Shack TRS-80 Color Computer is one of the most popular and versatile 6809-based systems to date. This article outlines this system and highlights a few of its more interesting features.

John Steiner
508 Fourth Avenue NW
Riverside, ND 58078

When I left the Radio Shack store with my new TRS-80 Color Computer, I wasn't aware that I was carrying one of the most advanced 8-bit microprocessors available. A personal computer has been on my want list for a long time, but until the Color Computer came along, I didn't feel I could afford one. I wanted a full-feature BASIC system, rather than a PC board, and the Color Computer met those requirements. Radio Shack is marketing this machine as a home and personal computer, rather than a business or industrial system. However, the 6809 and its multiprocessing capacity should make an excellent smart terminal, with the right software.

After having some exposure to the 6800 and programming in assembly language, I was pleased to find that the 6809 is upwardly compatible. However, this is at the mnemonic level only, and the 6809 requires opcodes that are different from the 6800. For an excellent reference to the 6809, read *6809 Microcomputer Programming and Interfacing* by Andrew C. Staugaard, Jr., Howard W. Sams & Co., Inc., 1981.

Hardware

The Color Computer system uses the E version of the 6809. The 6809E does not have an on-chip clock. Addressing and RAM refresh is handled by an LSI dynamic RAM controller chip. The Motorola 6847 video display generator IC is used as screen display through latch buffers. This display is fed to an internal video modulator, which also receives sound from one of four sound sources. (More on this later.)

There are two 6821 Motorola PIA's whose functions are: cassette interface (1500 baud), RS-232 interface (600 baud programmable to other rates), joystick interface, and keyboard interface. The keyboard uses calculator-style keys, and there is a 40-pin cartridge connector (that Radio Shack calls a ROMport) included. This ROMport makes available to external devices every major input and output line on the 6809. Maximum RAM capacity is 32K, though the basic machine comes with only a 4K by 8 chip set installed. Extended Color BASIC requires the 16K by 8 chip set. The Extended Color BASIC machine is a completely integrated system that is designed for the computer beginner (read average consumer) and the manuals and ROM-based software help the novice to become a hard-core "hacker."

Table 1

PMODE #	GRID SIZE	COLOR MODE
0	128 by 96	two color
1	128 by 96	four color
2	128 by 192	two color
3	128 by 192	four color
4	256 by 192	two color

Table 2: List of Graphics and Audio Commands

AUDIO	PCOPY
CIRCLE	PLAY
COLOR	PMODE
DRAW	PPOINT
GET	PRESET
LINE	PSET
PAINT	PUT
PCLEAR	SCREEN
PCLS	SOUND

Firmware

There is a standard 8K ROM-resident BASIC that could be (comparatively) called more advanced than Radio Shack Level I BASIC, but less advanced than Level II. The optional Extended Color BASIC requires an additional 8K ROM chip, and is roughly comparable to Level II BASIC. This adaptation, written by Microsoft, is graphics- and sound-oriented, though it has several string handling and programming statements and functions.

This standard floating point BASIC comes with trig functions and square root. Programming aids include a full-feature line edit, and a renumber statement that completely renumbers your BASIC program starting with any number specified, and incrementing by any number. This powerful command also renumbers GOSUBs and GOTOs, and identifies any undefined lines. A USR function allows machine language software to be executed from BASIC.

Another feature is a software-resettable timer. String and data handling capabilities include cassette files, and the TRS-80 graphics characters, which can be set to any one of the eight available colors. These characters are manipulated and stored like text, but there is no provision for single key entry of the characters. You must place them into your program as CHR$ statements.

Microsoft's Extended Color BASIC brings high-resolution graphics capability to the programmer. There are five modes that the color computer displays. Listed in table 1, are the five graphics modes. Any time text is output to the screen by the computer, or whenever

the programmer uses a print or input statemenr, the compurer automatically enters the text mode. This represents one of the few shortcomings of the Color Computer : text and high resolution graphics are not available on the same screen.

To write text on the graphics screens you must "draw" the words on the screen. The highest resolution available is 256 by 192, requires a minimum of 4 1.5K "pages" of memory, and is a two-color mode. The only colors available in this mode are black and green or black and buff. There are two medium and two low-resolution modes.

In PMODE 3 there are two four color modes available. Depending upon the screen set chosen, you can select from green, yellow, blue and red; or buff, cyan, magenta, and orange. This mode sets two points at a time, rather than one as in the high-resolution mode. The medium and low-resolution modes have both a four- and two-color mode, and the Lo-Res mode sets four adjacent dots to the same color. All points are plotted on a 256 by 192 grid so you have the option of choosing the mode which offers the best compromise between resolution and color requirements without having to change line, draw, or circle plots.

The video display generator requires 1.5K blocks or pages of RAM to store in graphics data, and the higher the resolution, the greater the memory requirements. The programmer can reserve up to eight pages for graphics use, and can switch from one page to another by software command. This allows a limited form of animation, bur there is a more versatile animation technique which will be explained later.

If the program that you are working on requires little or no graphics, you can reserve only one page for graphics and use the extra memory for your text program. Memory available for text at power up is 8.487K. By inputting a PCLEAR 1 command, you reserve one page for graphics and then 13.095K of memory for text programming. The remainder of the 16K is used by the computer for processing, as well as 200 bytes which are automatically cleared for handling strings.

While the compurer accepts a SET command SET(X,Y,C), where XY are the coordinates, and C is the color number, there is a much faster way to draw objects on the graphics screen. The LINE(X,Y)—(X,Y),PSET will draw a line of an earlier specified color from the left XY coordinates to the right XY coordinates. If the specified line is diagonal, and the PSET is followed by a ,B then the line is drawn as a rectangle with the leftmost XY as the upper left corner of the rectangle, and the rightmost XY as the opposite corner. One more option: an F after the B, fills this hox with the pre-specified foreground color. The left XY coordinates are optional also, and if left out, the computer starts drawing from the last specified XY point, or the center of the screen if none other had been previously specified.

Another useful graphics command is CIRCLE. The syntax for CIRCLE is CIRCLE(X,Y),R,C,HW,START,END. XY is the center of the circle, and R specifies its radius. All the rest are optional. C specifies its color, and HW specifies the height/width ratio of the circle thus making it into an elipse. START and END specify beginning and ending points for drawing an incomplete circle, as in making an arc.

PAINT (X,Y),C,B will paint the graphics sceen with color C, starting at XY and ending at the border of color B.

Probably the most versatile graphics command is DRAW. It specifics a string of characters that allow the programmer to manipulate graphics with string functions, which may be constants or variables. For example, A$="BM128, 96;U25R25D25L25": DRAW A$ executes a square which starts at 128,96, goes up 25 points, right 25 points, down 25 points, and left 25 points. The "BM" means blank move, and this moves the computer's last coordinate position to the new one without displaying the line on the screen. The start point can be made relative by stating "BM+25,+25" which starts drawing 25 units over and 25 units down from the last specified point. This feature is handy for creating graphics text strings to DRAW on the display. There are angle and scale options, a color specifier, a no update, and blank line option as well.

Two other useful graphics commands are GET and PUT. These graphics options set up a double dimensioned array that stores the particular colors of the specified screen location. For example, GET the rocket at the lower left corner of the screen, and PUT it at a higher location, then PUT it at a higher location, etc. There are options that allow you to AND or OR the array with the new location display. This technique is the fastest form of animation available to the Color Computer.

The audio capabilities are quite versatile in the computer. The basic 4K machine has a SOUND F,D command that sounds a tone of frequency F for a duration of D. Frequency and duration can be any number between 1 and 255, inclusive. Extended Color BASIC adds to this a PLAY command that allows the user to program the notes as written directly from a sheet of music in string format. There are several options, including tempo, length of note, pause, octave (five available), volume, and execute a substring.

I mentioned earlier that there were four sound sources. These include, a 6-bit D/A converter, a single bit sound source, which requires reprogramming the PIA, external sound from a ROM-PAK, or audio from cassette. Audio appears at the TV speaker, not from an internal speaker, as on the Apple. This necessitates that audio level be set at the TV, though the PLAY command can vary the volume of music, as long as the volume is up at the TV.

The TRS-80 Color Computer has a lot of power for the dollar, and with rhe 6809 microprocessors ability to use position-independent code, and its multiprocessing capability, there will be a degree of software compatibility heretofore not seen in the industry. Also terminal and Teletext applications will be more easily implemented.

Color Computer Cassette Interfacing

Editor's note: If you want to use your own cassette recorder with your Color Computer, your Radio Shack salesman may tell you you can't do it, since the only way to get the cable you need is with the Radio Shack cassette machine. They don't sell the cable or the connector separately!

The connector is available, however, on the printer end of the "Interface Cable" (part # 26-3009—$4.95) for the Quick-printer II. Just cut the cable in half and add extra wires and appropriate connectors for your cassette machine on the other end. You can then use the other half of the cord to make your own (non-Radio Shack) printer interface cable. The pin-outs for these connectors are available in the back of the "TRS-80 Color Computer Operation Manual" (pp. 26-27.)

The connectors, by the way, are standard DIN 4- and 5-pin connectors and might be obtainable at another electronics store.

MICRO

A TEAM OF 6809 SUPERSTARS:

Smoke Signal's Chieftain™ Computer, and Software by Microware

HERE'S THE *TOTAL* 6809-BASED SYSTEM FOR THOSE WHO DEMAND UNSURPASSED POWER, FLEXIBILITY AND RELIABILITY

After years of worldwide use in diverse and challenging applications, the outstanding performers in 6809 computer operations are SMOKE SIGNAL and MICROWARE. These leading companies are recognized as the undisputed choices **when there is no room for compromises.**

WHY SMOKE SIGNAL AND MICROWARE LEAD THE 6809 FIELD

Smoke Signal began pioneering research and development on 6800/6809-based computer systems back in 1977. Microware worked three years to perfect OS-9 and BASIC09.

Both companies have evolved outstanding 6809-based products from early engineering research, **and both pay almost fanatical attention to detail.** For example . . .

SMOKE SIGNAL'S 6809-based Chieftain™ computer series has **proven** its superiority in hundreds of demanding tasks. From gold-plated connectors to highest-quality materials throughout, each Chieftain™ is built to deliver absolute dependability from day one, and **stay** that way through years of service.

• ENDURANCE-CERTIFIED

Every Chieftain™ is meticulously ENDURANCE-CERTIFIED at 2.2 MHz. That's SMOKE SIGNAL's endorsement of product perfection.

MICROWARE's state-of-the-art OS-9 UNIX*-like operating system and the BASIC09 language have been developed in close coordination with computer manufacturers to maximize optimum system performance. The finest possible support and documentation further ensure satisfaction. Microware software performance is best summed up in this remark by a 25-year computer veteran:

"BASIC09 IS THE FINEST HIGH-LEVEL LANGUAGE I'VE EVER SEEN IN THE INDUSTRY!"

Thousands of engineers and programmers use MICROWARE software products as their standard time-saving tool . . . to execute process-control applications . . . and for other vital functions. COBOL and PASCAL are also available under the OS-9 operating system.

HOW THIS REMARKABLE TEAM OF COMPUTER SUPERSTARS CAN SERVE *YOU*

SMOKE SIGNAL's Chieftain™ computer provides an array of configurations ranging from 5¼-inch drives for single-user applications to multi-user, multi-tasking capabilities. Winchester hard-disk drive systems are also available.

In other words, **breathtaking power** with as little as 48k memory; Microware's OS-9 Level Two can access up to one full megabyte that your Chieftain™ can address!

One more sampling of the awesome processing potential at your fingertips with the Smoke Signal Chieftain™ computer:

MICROWARE'S Stylograph screen-oriented word processing package instantly makes Chieftain™ an easy-to-use document preparation system with comprehensive editing commands.

THERE'S MUCH, MUCH MORE! Call or write SMOKE SIGNAL for details on Chieftain™ computers and MICROWARE software.

SMOKE SIGNAL Dealer opportunities are still available . . . please request information.

*UNIX is a trademark of Bell Telephone Laboratories.

SMOKE SIGNAL BROADCASTING | MICROWARE

31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL (213) 889-9340

☐ Send information about Chieftain™ computers and Microware software.

☐ Provide information about Smoke Signal's Dealer program.

Name ______________________

Address ______________________

City __________ State ______ Zip ______

Telephone ( ) ______________________

Applesoft Mystery Parameter

This zero page location in Applesoft can be used to prevent the listing of a program, and for automatic "load and run" of tapes.

Sherm Ostrowsky
291 Salisbury Ave.
Goleta, California 93117

There is an undocumented booby trap deliberately hidden in Applesoft. A "soft switch" lurks in zero page location $D6 (decimal 214). If you POKE an appropriate number into this address, your Applesoft program will suddenly become inaccessible to you! You will still be able to run the program, but that's all you'll be able to do. No matter what valid or invalid Applesoft command you try to type in, the computer will interpret it as a "RUN" command and will restart the program from the beginning. You can neither list nor change the program. If you have the Autostart ROM, you can't even escape the trap by hitting RESET; this just puts you back into Applesoft, and any attempted command starts the program running again. When I first discovered this "switch" and tried it out, I finally had to turn off my Apple (thereby losing the program) before I could get out of this devastating situation.

Although I read almost all of the personal computer magazines, I have never seen anything written about this powerful, but hidden, feature of Applesoft. There is not a word about it in the various Apple reference manuals. In fact, the zero page memory map in the Apple II Reference Manual indicates that byte $D6 is not used by Applesoft, thereby implying that it is free for use by other machine language programmers hungry for space on page zero. Well, if you did try to use this particular piece of real estate as a handy storage spot in your ML program, you might be in for a bit of a surprise.

I first became aware of this situation when I read a remark made by W.E. Dougherty in his valuable compendium *The Apple Monitor Peeled*. He stated that in an Applesoft TAPE-SAVE, the first record is three bytes long; the first two bytes give the length of the Applesoft program to follow, and the third byte is always set to the number $55. But why are there three bytes, when two are sufficient to specify the length of any possible Apple program?

To find out about this third byte I began browsing through the disassembled ROM code. There I learned that the third byte was taken from memory location $52 just before the TAPE-SAVE operation began, and that after a TAPE-LOAD operation, this byte, which had been in $52, was deposited in memory location $D6. Why was such an elaborate process provided to transfer a single byte from one zero page location before a tape-save to another zero page location after the tape-load, if the byte in question was always set to $55?

It was then that I recalled having seen $D6 referred to at a critical point in the Applesoft routine which examines every input line and converts valid keywords into tokens for later implementation. Looking at this routine more carefully, I saw that if the number in location $D6 had bit seven set (i.e., it was greater than $7F or decimal 127), the keyword interpretation routine would be short-circuited. Without even looking at any keyword, the routine instead jumps unconditionally to a command sequence, which is equivalent to CLEAR and RUN! So that is how the "switch" is implemented: if you put into location $D6 any single-byte integer greater than $7F, the switch will be activated.

Suppose you have just written an Applesoft program and are preparing to SAVE it onto tape. If you type

```
POKE 214,128
```

you will never be able to SAVE the program at all. You will be trapped by this switch. But if instead you were to type

```
POKE 82,128
```

you'll have no such trouble. The switch remains off, and you can list, alter, and eventually SAVE your program. But if you LOAD this tape you will not be so fortunate. Your switch will be on from the instant the tape leader was loaded — even before the program itself is read into the computer. The only thing you will be able to do with the program is run it.

When activated by the third byte of a tape leader, the switch has yet another trick up its sleeve. The tape has become an "auto-run." It starts running as soon as loading is complete, without waiting for any input from the keyboard. This is consistent with the other activities of the switch. All of its efforts are directed toward preventing anyone from looking at, listing, or in any way meddling with the program. Naturally, to do this most effectively it would be necessary for the tape to auto-run. Otherwise, the user might be able to use the time after loading, but before typing RUN, to find a way of getting into the program.

Well, I can think of circumstances under which a fool-proof auto-run program tape would be a pretty useful thing to have. For example, if it is to be used by people who know nothing about computers, it would be best if they have as little to do with it as possible. A load-and-go tape will provide less chance for something to go wrong. And you cannot doubt that another reason behind this gimmick is the idea of protecting proprietary program details.

All right, then, how good *is* the protection afforded by this switch? Can it be easily circumvented? If so, can a program be reinforced so as to defeat such circumvention? Obviously, it is not possible to give a final answer to

these questions. Yet some remarks can be made concerning the operation of the D6 switch as it has been described thus far.

First of all, if you have the old Monitor ROM you can defeat the switch by simply hitting RESET. In this version of the Apple computer, hitting RESET invariably dumps you into the system monitor, recognized by the asterisk prompt: *. Once in the monitor, you are free from the effects of the switch. As far as I have been able to ascertain, this switch only functions in Applesoft. From the Monitor, therefore, you can type D6 (RETURN) and the system will respond by showing you the contents of this location. If the switch is on, the contents will be a hexadecimal byte between $80 and $FF. Now you only need to type D6 : 55 (RETURN) to turn off the switch (instead of $55, you can use any hexadecimal number less than $80 if you wish), and return to Applesoft by typing CONTROL-C. Now the program will behave normally. You can list it, change it, or do anything you want with it.

A careful examination of the listing of the monitor ROM in the *Apple II Reference Manual* has not shown me any way that this RESET fix can be defeated. The fact that RESET will always cause a jump out of Applesoft into the monitor seems to have been cast in silicon in this ROM version. It would take a hardware change to alter it. However, if I am wrong, I hope an alert reader will let me know about it!

The Autostart ROM is far more common than the monitor ROM in those Apples (especially the Apple II Plus) with Applesoft in ROM. As we have seen, hitting RESET in the autostart ROM usually dumps you right back into Applesoft with the program and all variables (including, of course, the contents of $D6) unchanged. So you will be no better off than before. However, with the autostart ROM, it is possible to make a *software* change which will simulate the operation of the monitor ROM (i.e., RESET will place you in the monitor). This function is described in great detail on page 37 of the *Apple II Reference Manual*. Just remember that as soon as you turn on your Apple (before you try to LOAD the protected program), execute the following commands in immediate mode:

```
POKE 1010,105 : POKE 1011,255
: CALL -1169
```

From then on, RESET will return you to the monitor, and you may be able to defeat the switch as described above. However, *this* fix is *not* cast in silicon — software made it, and software can overrule it. All the protected program needs as one of its early statement lines, is something like this:

```
10 POKE 1010,102 : POKE
1011,213
: CALL -1169
```

Now, after you've fixed the RESET key to simulate the monitor ROM, you must hit RESET in the fraction of a second between the completion of the tape LOAD and the beginning of the program's auto-RUN. The program line listed above undoes software fix on the RESET operation and even makes the situation worse. Now hitting RESET not only leaves you in Applesoft, but also jumps to the RUN command and starts running the protected program from the beginning again!

The alert reader has probably noticed one glaring omission from this article so far. I've said a lot about the TAPE-SAVE and TAPE-LOAD procedures, and about protected and auto-run tapes, but nothing about disks and DOS. This is because my system does not yet have a disk, so it has not been possible to check out the situation in DOS. Nevertheless, it seems unlikely that such a clever and useful gimmick would have been omitted from the DOS. This will be investigated as soon as I can get access to a disk system.

One more possible implication may be worth pointing out. Applesoft was written by the same Microsoft people who have produced versions of BASIC for several other computer systems. It seems likely to me that they might have incorporated a similar switch into some of these other versions, too. So you PET owners who are handy with 6502 assembly language might want to try hunting for it somewhere in one of the various different PET ROMs.

Sherm Ostrowsky, a physicist, has been working with computers since 1958. He spends much of his time exploring non-conventional computer architectures. Ostrowsky owns an Apple and is fluent in 6502 assembly language.

MICRO

PRICE BREAKTHROUGH

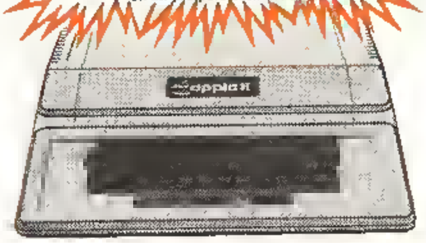
We've done it again! As a leader in small computer sales we must constantly find new ways to save customers money while offering quality products. Our most popular seller, the Apple II Plus, is now available with 64K of user RAM at the unbeatable price of only $1249 This is possible because we manufacture the 16K RAM Card that expands the factory Apple II 48K to it's maximum capacity of 64K. The Ram Card allows use of Integer and Applesoft Basic and other languages like Pascal. It's a must for large data bases, Visicalc, and the Z-80 cards. The card is made from high quality components and has a full one year warranty.

64K APPLE II PLUS*

**48 K Apple II Plus with 16K Ram Card*

ONLY $1249

FREE SHIPPING*
*on all pre-paid cash orders



16K ONLY **$1025** **48K** ONLY **$1089**

OISK II DRIVE WITH CONTROLLER CARD **$499**

OISK II DRIVE ADD ON **$439**

Apple Cards and Hardware

16K Ram Card by CCI	130
Language System w/Pascal	379
Silentype Printer w/Interface Card	349
Hayes Micromodem II	299
Novation Apple-Cat II	339
Videx Videoterm 80 Column w/Graphics	269
Z-80 Softcard by Microsoft	299
16K RamCard by Microsoft	159
ABT Numeric Keypad (old or new kybrd)	110
ALF 3 Voice Music Card	239
ALF 9 Voice Music Card	169
Lazer Lower Case Plus +	55
Micro-Sci Disk Drives (A-40 & A-70)	CALL
SSM AIO Serial/Parallel Card A&T	189
Sup-R-Terminal 80 Column Card	329
SVA ZVX4 Megabyter 8" Disk Controller	589
SVA 2+2 Single Den. 8" Disk Controller	345
ThunderClock Plus	119
Symtec Hi-Res Light Pen	210
Integer or Applesoft Firmware Card	145
Graphics Tablet	619
Parallel Printer Interface Card	135
Hi-Speed Serial Interface Card	135
Smarterm 80 Column Card	299
Joystick by Keyboard Co.	45
Music System (16 Voices)	479
A/D + D/A Interface	289
Expansion Chassis	599
Introl/X-10 Controller Card	169
Clock/Calendar Card	225
CPS Multi-function Card	189
Supertalker SD-200	239
Romplus + Card	135
Romwriter Card	149
Clock/Calendar Module	99
GPIB IEEE-488 Card	249
Asynchronous Serial Interface Card	129
Centronics Parallel Interface Card	99
Arithmetic Processor Card	299

We carry all CCS cards, please call for best prices.

Software for the Apple

Visicalc 3.3	169
CCA Data Management	85
DB Master	169
WordStar (Apple 80 col. version)	299
Applewriter	65
Easywriter	225
Peachtree Business Software	CALL
Visiterm	129
Visitrend/Visiplot	219
Real Estate Analyzer	89
Tax Preparer	99
DOS Toolkit	65
Tax Planner	99
Dow Jones Portfolio Evaluator	45
Dow Jones News & Quotes Reporter	85
Apple Fortran	165
Controller Gen. Bus. System	499

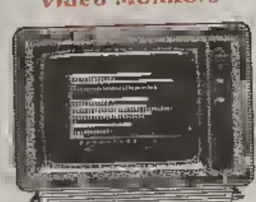
Printers

Epson MX-80 CALL

Anadex 9500/9501 w/2K Buffer	1299
C. Itoh Starwriter 25 CPS	1499
C. Itho Starwriter 45 CPS	1899
Epson MX-70	CALL
Epson MX-80 F/T	CALL
Paper Tiger IDS-445	699
Paper Tiger IDS-460	949
Paper Tiger IDS-560	1249
Silentype w/Apple II interface card	349
Quine Sprint 5/45	2495

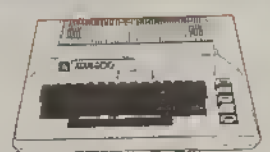
TOLL FREE ORDER LINE
800-854-6654
California and outside Continental US
(714) 698-8088
TELEX 695000 BETA CCMO

Video Monitors



Amdek/Leedex Video 100 12" B&W	139
Amdek (Hitachi) 13" Color	359
NEC 12" Green Phospher Screen	CALL
NEC 12" RGB Hi-Res Color	CALL
Panasonic 13" Color	449
Sanyo 9" B&W	185
Sanyo 9" Green Phospher Screen	CALL
Sanyo 12" B&W	269
Sanyo 12" Green Phospher Screen	285
Sanyo 13" Color w/excellent resolution	449

Atari Personal Computer



ATARI 800 16K $749

Atari 400 16K	349
410 Program Recorder	65
810 Disk Drive	449
815 Dual Disk Drive (Dbl. Den.)	1295
822 Thermal Printer	349
850 Interface Module	159
16K Ram Memory Module	149
16K Ram Memory Module (by ConComp)	89

NEC Microcomputer

PC-8001A 32K Ram Computer	CALL
PC-8012A I/O Unit w/32K Ram	CALL
PC-8031A Dual Mini-Disk Drive Unit	CALL

Please write for more information about the NEC computer.

Ordering information: Phone orders using VISA, MASTERCARD, AMERICAN EXPRESS, DINER'S CLUB, CARTE BLANCHE, bank wire transfer, cashier's or certified check, money order, or personal check (allow ten days to clear). Unless prepaid with cash, please add 5% for shipping, handling and insurance (minimum 5.00). California residents add 6% sales tax. We accept CODs OEM's, Institutions and corporations please send for a written quotation All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty (usually 90 days) Showroom prices may differ from mail order prices

Send Orders to:

consumer computers Mail Order

8314 Parkway Drive
La Mesa, California 92041

MICRO

From Here to Atari

James Capparell
297 Missouri
San Francisco, California 94107

My purpose in writing this column is to give potential microcomputer users enough information on the Atari 800 to provide a basis of comparison to other micros. I want to point out the features that convinced me that the 800 is perhaps the most-for-your-money machine on the market. I also think it is the best graphics machine in its price range. The smaller 400 unit, while possessing a difficult (for touch typists) keyboard, has so many of its bigger brother's features that I feel it may be the single best buy in the entire computer industry.

The following will apply to both 400 and 800 units unless noted.

When you look closely at this equipment, you'll notice that the design engineers went to considerable effort to put together a system that is sophisticated and versatile, yet easy to use for the novice. This system is modular and therefore easily expandable. The designers even built the TV modulator into the chassis. This device allows you to connect your television to your computer. Many manufacturers force you to pay extra for the modulator even though it's absolutely essential for proper connection to your home television.

A second video output is available (800 only), allowing the connection of another monitor. This second port provides composite color/luminance and therefore a generally better picture. The two pictures are handy in teacher/student situations or any place where two pictures are better than one.

While on the topic of video, Atari puts out a clean NTSC (National Television Systems Committee) standard video signal suitable for home video taping. This is accomplished via the DIN plug (800 only) on the side of the console. Other machines need special adapter boards to do the same thing.

Atari was one of the first companies to pass FCC RFI emission standards. You can safely use this machine at home and not fear your neighbor's wrath when he can't watch his favorite television show due to interference from your computer. Many manufacturers had to redesign equipment to meet these stringent requirements, at an added cost to buyers. Again, Atari engineers demonstrated their design foresight.

Another video feature not readily apparent to the casual observer is the allowance for misaligned televisions. By including a border around the displayable screen image, Atari engineers have assured us that when we use VisiCalc the numbers won't disappear into the invisible portion of the screen. This border is under programmer control.

One last design feature that even many Atari owners don't yet appreciate is something euphemistically called ATTRACT MODE. This is that strange habit that causes the Atari to begin to rotate through all of its colors when no one has pressed on the keyboard in the previous nine minutes. This feature was included to prevent any one phospher from burning out.

I can't speak about this equipment without mentioning graphics. The incredible features are due to a trio of large-scale integrated circuits designed specifically by and for Atari. These chips make twenty different graphics/text combinations available (depending on memory available). The high-resolution mode is 320 horizontal dots by 192 vertical lines. Actually, there is a way to get more than this (recall the border), but I'll leave this technique for a future column.

There are two channels of DMA (direct memory access) video available. There are 128 color luminance combinations and all may be on the screen at one time. To be fair, there are constraints when using so many colors and the techniques will probably only be used by intermediate to advanced programmers. Also available are priority and collision registers that allow advanced animation techniques to be used.

The character set can be completely redefined by the user. This will allow creation of such things as foreign alphabets or special mathematics characters. Available to advanced users is hardware-controlled smooth scrolling in any direction, display list-driven playfield graphics, display list interrupts — all advanced techniques suitable for future columns.

Standard on these machines is a set of parallel ports usually referred to as the front jacks. These are ports A and B of a 6820 PIA. They allow the connection of a variety of input devices. Light pen, barcode reader, graphics tablets, and printers all have been implemented using these jacks. Also, the standard joystick and paddle controller plug in here. In addition, there is a serial port included to which a printer, disk, cassette, and modem may be added. To include all these peripherals, you must purchase an I/O expansion unit.

The console comes complete with an upper/lower case keyboard with a feel that touch typists seem to like. (Note that the 400 has a flat keyboard more appropriate for youngsters.) With this keyboard is an easy-to-use screen-oriented editor with full cursor control. This makes BASIC, assembler FORTH, Pascal or Pilot easy to enter and correct.

The 400 and 800 units also have four voices, or channels of sound, available. This allows you to play four-part harmony, or to include special sound effects in your latest program. The sound feature is readily available in BASIC. I've heard everything from Beethoven's *Fifth* to the latest in phaser sounds on my machine.

In conclusion, I'd recommend the Atari 800 to anyone really serious about using state-of-the-art graphics. This includes all game players, as well as experimenter/hacker types. Also, those of you interested in experimenting with video taping must consider this machine. For potential users looking for business software, I've seen and heard of some excellent packages to be released soon. These include a full general ledger system, word processor and data base system for starters. I would not hesitate

to suggest this equipment to those of you who would like to have a very flexible tool without becoming professional computer programmers. From the standpoint of ease-of-use, its modular design, and of course its state-of-the-art components, it's a machine you can live with.

The 400, since it has almost as much capability as its bigger brother, must be seriously considered, and in fact may be the only machine to consider in the under $500 price range. If you are not sure about your needs and don't want to spend much, yet still want all the frills that the rest of us have come to expect, the 400 is perfect.

I hope I've managed to convey the excellent combination of features and price. More of us are discovering this and my intent here is to raise issues of comparison. Now, when your friendly dealer attempts to sway you to purchase some other equipment you will be able to do some intelligent comparison shopping. (Please see summary table for these features, as well as a few additional ones.)

I intend to be doing a regular column for MICRO. Some of these advanced features will make excellent articles. I welcome suggestions and questions.

Summary Table: Atari Features

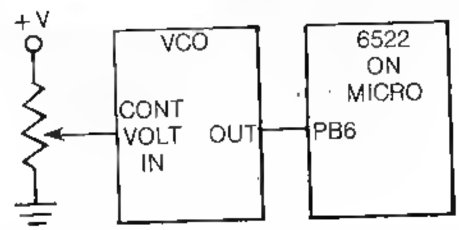
1. Full screen editor
2. Full upper/lower case typewriter-like keyboard
3. Full cursor control: up, down, left, right
4. Complete RFI shielding
5. Built-in TV modulator
6. NTSC standard signal, suitable for video taping
7. Second video output (available on the 800)
8. Protection against phospher burn on home TVs
9. 128 color/luminance combinations available
10. Twenty different graphics/text combinations available
11. Maximum resolution of 320 × 192
12. Two DMA video channels available
13. Capability of placing eight small objects on screen all with independent color, position, and priority control registers (known as player/missiles)
14. Hardware-controlled smooth scrolling
15. Built-in real time clock and four additional counters
16. Display is list-driven, bit-mapped video (known as playfield graphics)
17. Interrupt-driven system with many hooks available to programmer (*e.g.* vertical, blank, or scan line interrupts are accessible)
18. Four channels of sound are standard with frequency, volume, and distortion control in BASIC. Envelope shaping is possible from machine level code
19. The separate voice channel available on cassette recorder is suitable for educational purposes
20. Parallel port built in
21. Serial port built in

MICRO

6522-Based Pot Position Digitizer

This article describes a simple method of converting the position of a potentiometer into a number that can be used by tha programmar for control applications or games. The method uses the pulse-counting capability of a 6522 and almost any inaxpensive VCO.

Figure 1: **Block diagram of a simple Pot Position Digitizer (PPD).**



Kenny Winograd
Seven Greenbriar
N. Reading, Mass. 01864

This article will detail a basic method of digitizing the position of a pot (potentiometer). A Pot Position Digitizer (PPD), in its most general terms, is simply a device which converts the position of a pot to a number which can be used by a computer program.

Many quantities we deal with every day are often set by the turning of a knob, not by the inputting of an actual number. Examples include the volume setting on your stereo, the setting of a lamp dimmer, a steering wheel, the position of a paddle in a TV game, etc. Knowing that a quantity represents a minimum amount may be more important than knowing the exact quantity.

You may want a PPD that gives the number $00 if the pot is turned fully counter-clockwise and the number $FF if the pot is turned fully clockwise. This number may represent anything from the initial fuel on-board in a lunar lander game to the number of microseconds to remain in a delay loop. (Numbers that follow a dollar sign -$-indicate hexadecimal numbers.)

A block diagram of the system is shown in figure 1. It is simply a variable voltage from a potentiometer feeding a voltage controlled oscillator (VCO) which feeds a single input pin of a 6522 Versatile Interface Adapter chip (VIA). I've chosen the single input pin to be PB6 so I could take advantage of the external pulse counting capability of this pin on a 6522. (More on this later.)

The whole idea is this: first, write a short program that counts the number of pulses entering PB6 in, say, 10 milliseconds. Then adjust the VCO so that when the pot is at its maximum position, the VCO will send out 255 pulses in 10 milliseconds. When the pot is at its minimum position, the VCO doesn't send out any pulses. That's it. Every time we use this code (which I've made into a subroutine), we are left with a number which represents the position of the pot.

This article will introduce you to a simple VCO circuit, as well as take you through a bit of software. The software will be used to "read" (actually, count) the pulses produced by the VCO. Now let's get to some specifics.

Hardware

A VCO feeds pulses to the microcomputer. (VCOs are also called Voltage to Frequency Converters or V to Fs — same thing.) In order to get 255 pulses in 10 milliseconds, we'll need a top frequency of at least 25500 Hz. To give us a bit of leeway, let's say we try for 26000 Hz. In this way we can be assured that when the pot is at its maximum position, in 10 milliseconds we will be able to count just a bit more than $FF pulses. Otherwise you may turn the pot to maximum and never get to $FF. $FE might be the highest number obtainable.

We don't want to go too much higher than the calculated value of 25500 Hz, because we would end up with a large dead band. A dead band would mean that when adjusting the pot upward, we would reach $FF and still have more rotation to go with the number remaining unchanged. As we'll find out later, this won't be much of a problem. I've included some software that will make calibration of the VCO simple.

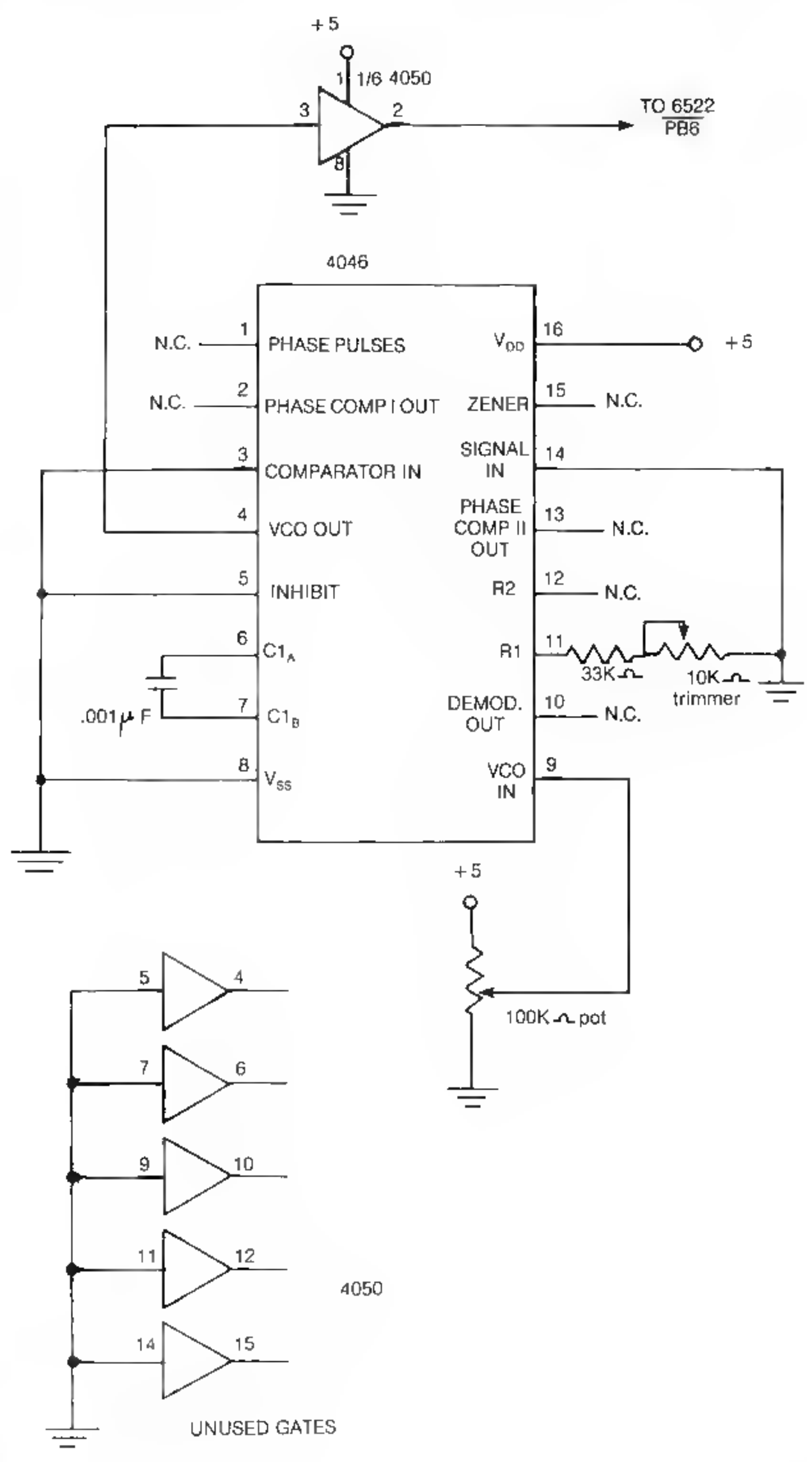
Figure 2 shows the schematic of a VCO based on the CMOS chip 4046. The 4046 will run on +5 volts and requires virtually no current. This chip also incorporates a couple of phase detectors, a zener diode and more. These extra circuits on the chip we either leave disconnected or properly terminated.

With the pot at its minimum position (i.e. ground) the VCO produces no pulses... a frequency of 0 Hz. With the pot at its maximum position (i.e. 5 volts) the VCO produces its maximum frequency. This maximum frequency is determined by two things: the value of the capacitor connected between pins 6 and 7 of the chip, and the value of

resistance from pin 11 to ground. The components shown, permit a maximum frequency of 26000 Hz. If you want to be on the safe side, or just experiment a little, you can change the resistance on pin 11 to be a 10K fixed resistor and a 100K ten-turn trimmer. Now you'll have a very large range of adjustment in the event of problems with capacitor tolerance. A .001 microfarad disk cap may actually measure out as a .002 microfarad cap, especially if you use old "junk-box" parts, which is fine, as long as we account for it. The equation for the maximum frequency of the circuit shown in figure 2 (with the pot turned up to maximum) is as follows:

$$f_{max} = \frac{1}{R(C+32pF)}$$

Figure 2: Schematic of a VCO based on a CMOS phase-locked-loop chip, a 4046.



If you feel like doing a little math (watch the units), you can verify that with the components shown (C = .001 microfarads and R = 38K with the trimmer at midpoint), the maximum frequency works out to be about 25500 Hz. If you do a little experimenting with this VCO, realize that even the CMOS data book states that this equation should only be used as a guide. It is also only valid if R is between 10K and 1M, and C is between 100 picofarads and .01 microfarads.

The 4046 VCO input resistance is very high, so virtually any value pot may be used. I've shown a 100K pot, but you can try anything from a 10K to a 1M pot.

Since the 4046 is a CMOS chip, we must make the output of the VCO compatible with the TTL input (PB6 on the 6522). I did this using a CMOS buffer. A 4049 or a 4050 may be used for this purpose.

Software

The code that I've written takes full advantage of the two interval timers within a 6522 VIA chip. Since I own a SYM-based microcomputer system, the code is perfectly compatible with a SYM. If you own any other 6502-based computer with a 6522, the code is again compatible as long as you change the addresses of the 6522 internal registers to fit your memory map. I used SYM VIA #2 (PB6 = pin AA-H).

The code is written in the form of a subroutine, and is fully relocatable and PROM-able. By PROM-able, I mean that there are no locations within the body of the subroutine that get written into, so that the code can reside in read only memory. The only internal registers in the 6502 used by the subroutine are the accumulator and the flag register, and these are preserved by the subroutine.

The basic functions that the POTDIG subroutine must perform are as follows: 1) set up the T2 timer to count externally input pulses on PB6, 2) set up the T1 timer to count to 10 milliseconds (in this example), 3) after T1 times out, determine how many pulses were counted on PB6, and 4) make this number available for use by the main program. That's basically all there is to it, except for one thing.

I've added a little code called OPTION. Here is where you can personalize the PPD to your heart's content. The code shown will only allow numbers from $01 through $FF. I chose to disallow $00 since if I use the PPD output number (stored in HEXLO) in a delay loop and do a decrement before a test for zero, a $00 decremented will become a $FF and I may go through a loop $100 times and not $00 times. Your option code may only allow numbers from $00 to $FE because of trouble with increment before test for zero instructions.

You may also want to scale the numbers differently. For instance, if instead of $01 to $FF, you wanted numbers in the range of $00 through $0F, you could take the PPD output number and shift it to the right four places. (1111 1111 would become 0000 1111; that is, a $FF would become a $0F.)

One more example of a different OPTION code would be if only numbers from $80 through $FF were wanted. Take the PPD output number, shift it one bit to the right (which divides the number by two), and add it to $80. This would result in numbers from $80 through $FF.

The subroutine POTDIG, after saving any registers that will be used, sets up bit 6 of port B (PB6) as an input pin. Then it puts PB6 (T2) in the external pulse counting mode by storing a one in bit 5 of the Auxiliary Control Register (ACR). Note that the same instruction (LDA #$20) also stored a zero in bits 6 and 7 of the ACR putting the T1 timer in the mode that generates a single interrupt. The next pair of instructions clears the T1 and T2 interrupt enables. The interrupt ability of the 6522 chip is not required in this example.

The next section of code loads the two timers, clears the T1 and T2 interrupt flags and starts the counting. The T1 counter is used to "gate" the external pulses to PB6. Actually PB6 will always be looking at pulses, but they will only be counted until T1 times out. I've loaded $270F into the latches of T1. If you recall, the timers actually count down and do not flag you until they count through zero, not at zero. Thus, T1 will not time out until $2710 microseconds. ($2710 microseconds = 10000 microseconds = 10 milliseconds.) I've loaded $FF into T2LL and T2CH. This count, $FFFF, will be decremented automatically by each incoming negative pulse on PB6. Later, after T1 times out, we'll calculate how far T2 has counted down to learn how many pulses arrived at PB6.

Listing 1

```
                  ;* POT POSITION DIGITIZER SUBROUTINE *
                  ;*                                   *
                  ;*       BY KENNY WINOGRAD           *
                  PBDD   EQU $A802        ;ALL ADDRESSES OF THE FORM $A8XX
                  T1LL   EQU $A804        ; REFER TO THE INTERNAL REGISTERS
                  T1CL   EQU $A804        ; OF THE SELECTED 6522 CHIP
                  T1LH   EQU $A805
                  T2LL   EQU $A808
                  T2CL   EQU $A808
                  T2CH   EQU $A809
                  ACR    EQU $A80B
                  IFR    EQU $A80D
                  IER    EQU $A80E
                  HEXLO  EPZ $00          ;FINAL POTDIG OUTPUT NUMBER LOC.
                  HEXHI  EPZ $01          ;HI-ORDER WORKING STORAGE LOC.
                  ;
                         ORG $300
                         OBJ $800
                  ;
0300 08           POTDIG PHP              ;SAVE FLAGS
0301 48                  PHA              ;SAVE ACCUMULATOR
0302 AD02A8              LDA PBDD         ;ENSURE PB6 IS SET AS INPUT BIT
0305 29BF                AND #%10111111   ;AND LEAVE OTHER BITS ALONE
0307 8D02A8              STA PBDD
030A A920                LDA #%00100000   ;SET PB6(T2) TO PULSE COUNT MODE
030C 8D0BA8              STA ACR
030F A960                LDA #%01100000   ;CLEAR T1 & T2 INTERRUPT ENABLES
0311 8D0EA8              STA IER
0314              ;
0314              ;NOTE: $270F+$0001=$2710 MICROSEC.=10 MILLISEC.
0314              ;
0314 A90F                LDA #$0F         ;LOAD T1 COUNTER WITH $270F
                                          ;CLEAR T1
0316 8D04A8              STA T1LL         ;INTERRUPT FLAG AND START COUNTING
0319 A927                LDA #$27
031B 8D05A8              STA T1LH         ;DOWN AT SYSTEM CLOCK RATE
031E A9FF                LDA #$FF
0320 8D08A8              STA T2LL         ;LOAD T2 COUNTER WITH $FFFF;
                                          CLEAR T2
0323 8D09A8              STA T2CH         ;INTERRUPT FLAG & GO COUNT PB6
                                           PULSES
0326 AD0DA8       WAIT   LDA IFR          ;CHECK IFR--HAS T1 TIMED OUT?
0329 2940                AND #%01000000   ;NO, KEEP CHECKING
032B F0F9                BEQ WAIT         ;YES, GO FIGURE # OF PB6 PULSES
                                           COUNTED
032D 38                  SEC              ;PREPARE FOR SUBTRACTION
032E A9FF                LDA #$FF         ;GET LO-ORDER COUNT AND
0330 ED08A8              SBC T2CL         ; CLEAR T2 INTERRUPT FLAG.
0333 8500                STA HEXLO        ;STORE LO AT HEXLO IN ZERO PAGE
0335 EA                  NOP
0336 A9FF                LDA #$FF
0338 ED09A8              SBC T2CH         ;FIGURE HI-ORDER COUNT
033B 8501                STA HEXHI        ;STORE IT IN HEXHI IN ZERO PAGE
033D EA                  NOP
033E AD04A8              LDA T1CL         ;CLEAR T1 INTERRUPT FLAG BEFORE
                                           LEAVING
0341              ;THE OPTION CODE THAT FOLLOWS ENSURES THAT ON LEAVING
0341              ;THIS SUBROUTINE, THE DIGITIZED POT COUNT WILL BE
0341              ;BETWEEN $01 AND $FF.
0341 A501         OPTION LDA HEXHI        ;TEST FOR GREATER THAN $FF
0341                                      ; BY LOOKING
0343 EA                  NOP
0344 F007                BEQ OPTA         ; FOR A NON-ZERO HI-ORDER BYTE
0346 A9FF                LDA #$FF         ; IF GREATER THAN $FF, SET TO $FF
0348 8500                STA HEXLO
034A EA                  NOP
034B D00A                BNE OKAY
034D              ;
034D A500         OPTA   LDA HEXLO        ;IF LOW-ORDER BYTE EQUALS ZERO,
034F EA                  NOP
0350 D005                BNE OKAY         ; SET EQUAL TO $01. IF NOT,
0352 A901                LDA #$01           LEAVE AS IS.
0354 8500                STA HEXLO
0356 EA                  NOP
0357              ;
0357 68           OKAY   PLA              ;RESTORE ACCUMULATOR
0358 28                  PLP              ;RESTORE FLAGS
0359 60                  RTS              ;RETURN TO MAIN PROGRAM
                         END
```

The small loop, starting at WAIT, keeps checking to see when the T1 interrupt flag is set in tbe Interrupt Flag Register (IFR). When set, it's time to find out how many pulses got through. Since we are dealing with a counter that starts at $FFFF and counts down, we subtract the numbers in the T2 counter from $FFFF, leaving us with the proper count. I've stored this 16-bit number in zero page locations $00 and $01, also known as HEXLO and HEXHI. These two locations may be changed to two other locations somewhere else in zero page if they conflict with other program use. After each STA ZERO PAGE instruction, I've put a NOP instruction just in case you want to use a different addressing mode to store these numbers at the top of your RAM. At this point in the subroutine, the T1 interrupt flag is cleared by reading the contents of T1CL.

Let's review a bit. If all we want to get out of this routine is a number from $01 through $FF, why do we need two bytes to store it in? Well, remember I recommended earlier that you set the maximum frequency of the VCO just slightly higher than the calculated frequency so as to be sure that you can get the maximum count by turning the pot to maximum. If we just used the low order counter of T2 and the VCO was a bit fast (on purpose or not), we may read a value of, say, $03 and not know that what we really have is $103. Now, in the code labelled OPTION, we test for just that kind of thing. If we find the high order byte to be anything other than zero, we know our final answer (in HEXLO), should be $FF. If we find the high byte is zero, then we work on the low byte (which we know must be between $00 and $FF). Here, as previously mentioned, I chose to ensure that the lowest number possible is $01, thus dis-allowing a $00. If I find a $00, I make it a $01

The OPTION part of the code is personalized for an application of mine. You should make it compatible with what you'd like. Restore the previously saved registers and return to the calling program.

I've assumed in the previous discussion that you are already, at least, somewhat familiar with the workings of the two timers on the 6522. If this not be the case, try reading the article by Marvin L. DeJong in MICRO (17:27) called "6522 Timing and Counting Techniques." It reads a lot easier than the 6522 data sheet.

One reminder... if you change POTDIG to use a VIA other than the one I used, be sure you connect the output of the VCO to the correct PB6 pin.

Calibration

If you own a frequency counter, calibration can be performed in about two seconds flat. Power up your VCO, connect the frequency counter to the output of the VCO, turn the PPD pot up to maximum and adjust the VCO trimmer until you read somewhere between 25500 Hz and 26000 Hz.

(continued)

If you don't have a frequency counter, use a simple program tbat repetitively calls POTDIG, gets the final result from HEXLO, displays it for a while and then jumps back to the beginning. Tbe program can be used for calibration. I've included the listing of a short program that does this for a SYM using some monitor subroutines (see listing 2).

When this code is executed, turn the pot back and forth and watch the displayed numbers go up and down. GO = $0200. This code will work with either the SYM 7-segment display or the video monitor or TTY. The method of display will, of course, appear differently but calibration is easily done either way. The procedure is to turn the pot up to maximum and adjust the VCO trimmer a little bit higher tban what is needed to get a solid display of $F's. All of a sudden your display device will become very stable... it will look less like garbage. At first, you probably must have thought that either you entered something wrong or I was nuts. If you're using the SYM 7-segment display, you'll always note a little flicker because every time POTDIG is called, about 10 milliseconds go by before the display is scanned again for a while.

MICRO

Listing 2

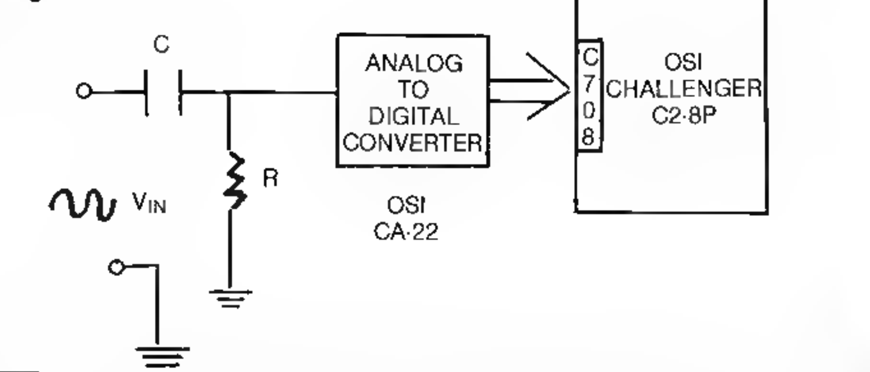
```
                   ;*******************************
                   ;*                             *
                   ;* A SIMPLE CALIBRATION AID--  *
                   ;*  MAIN PROGRAM FOR A SYM     *
                   ;*                             *
                   ;*      BY KENNY WINOGRAD      *
                   ;*                             *
                   ;*******************************
                   ;*
                   ACCESS EQU $8B86          ;UNWRITE-PROTECT SYSTEM RAM
                   CRLF   EQU $834D          ;CARRIAGE RETURN-LINE FEED
                   OUTBYT EQU $82FA          ;OUTPUT ACC. AS TWO HEX DIGITS
                   SCAND  EQU $8906          ;SCAN SYM 7-SEG DISPLAY
                   ;
                   POTDIG EQU $300           ;POT POS. DIGITIZER SUBR. LOC.
                   ;
                   TEMP   EPZ $03            ;TEMPORARY STORAGE
                   HEXLO  EPZ $00            ;POT DIG. NUMBER OUTPUT LOC.
                   ;
0800 20868B        CALIB  JSR ACCESS
0803 200003               JSR POTDIG
0806 A500                 LDA HEXLO          ;GET POTDIG OUTPUT #
0808 EA                   NOP
0809 20FA82               JSR OUTBYT         ;OUTPUT IT
080C 204D83               JSR CRLF
080F A9FF                 LDA #$FF
0811 8503                 STA TEMP
0813               ;
0813 200689        LOOP   JSR SCAND          ;DISPLAY IT AND DELAY FOR A WHILE
0816 C603                 DEC TEMP
0818 D0F9                 BNE LOOP
081A F0E4                 BEQ CALIB          ;GO BACK AND DO THE SAME THING
                        END                   AGAIN.
```

6502 Frequency Counter

Turn your microcomputer into en inexpensive frequency counter using en A/D converter. Demonstretion programs for OSI C2-8P ere provided.

Phil Lindquist
8892 Cooley Lake Road
Union Lake, Michigan 48085

Figure 1



The Problem

My personal computer for the past year has been an Ohio Scientific Challenger C2-8P with dual floppy disks and an analog I/O board. Recently, while cbecking the operation of a new modem circuit, a need arose for some simple audio test equipment.

This article describes one of the programs subsequently developed: a frequency counter witb a maximum tbeoretical count rate in excess of 10 KHz utilizing a 6502 microprocessor operating at 1 MHz. An assembly language listing of the actual frequency counting routine is provided, as well as a driver written in Microsoft BASIC that calls the counter as a USR(X) subroutine.

The Approach

The heart of the frequency counter is a machine code timed loop requiring 46 microprocessor cycles, which checks for positive signal transitions, and increments a counter whenever one is encountered. Tbe program cycles through the loop 21739 times (54EB H), which requires one second with a I MHz microprocessor clock. The trick is to keep all possible paths through the loop (lines 410 to 770) equal in length.

The duration counter keeps track of tbe number of loop cycles remaining. Upon entering the program, the count is set to 21739. Note that to compensate for variations in microprocessor clock speeds, this value may have to be changed.

You can input to the system througb an analog to digital (A/D) converter such as the new OSI CA-22 analog interface board, or the older OSI 430B analog interface board (no longer available). Tbe CA-22 analog I/O board is a high-performance interface tbat is capable of multiplexed 12- or 8- bit A/D conversion, using sample and bold and successive approximation techniques. In addition, the CA-22 will perform up to 66,000 8-bit A/D conversions each second. This capability is more sophisticated than is required for tbe frequency counter application, and alternative hardware implementations may include other A/D converters or even an op amp configured as a threshbold detector and driving a single bit input. However, the alternates will require minor modifications to the programs I will describe here.

The CA-22 as delivered from OSI is set to transform an analog voltage input in the range of -10 volts to + 10 volts into complementary offset binary (COB) digital representation. Complementary offset binary is summarized in table 1.

The transition from a negative voltage to a positive voltage is reflected by a change from one to zero in the most significant binary digit.

The 6502 Branch on MInus (BMI) instruction is conditional, depending on the state of the most significant bit of the tested byte. Therefore, using this instruction we may easily determine if the signal input to the A/D converter is greater than or less than zero volts. An alternative, if a TTL signal is being counted, is to connect the signal directly into the most significant bit of a parallel input port.

The bardware configuration using an A/D converter is diagrammed in figure I. The capacitor and resistor balance the signal input to the A/D around 0 volts.

Listing 1

```
10                ; ***********************************************
20                ; *                 Program   F R E Q
30                ; * OSI 6500 Assembler Code for Frequency Counter
40                ; * Program by P. Lindquist, Union Lake, Michigan
50                ; *                  January  1980
60                ; ***********************************************
70                ;
80                ;  Program monitors input and counts positive
90                ;  transitions occurring during a fixed number
100               ;  of cycles through the program as determined
110               ;  by count entered into DR.
120               ;
130               ;  EXTERNAL ADDRESSES AND VARIABLES:
140               ;
150 0000=          CNTL    = $0000        ;Frequency Count - lo byte
160 0001=          CNTH    = $0001        ;Frequency Count - hi byte
170 0002=          DRCL    = $0002        ;Duration Count - lo byte
180 0003=          DRCH    = $0003        ;Duration Count - hi byte
```

```
190 0004=          LAST   = $0004         ;Positive or Negative Signal
200 C708=          INPUT  = $C708         ;CA-22 A/D Input Address
210 C70A=          MUX    = $C70A         ;CA-22 A/D Control Address
220 0020=          MUXSET = $20           ;Multiplexer control, eight
230                ;                          bits, port zero, no flag
240 317E                 * = $317E
250                ;
260 317E EB54      DR     .WORD 21739     ;Duration count for 1 Sec
270 3180 0000      RES    .WORD 0         ;Return location for results
280                ;
290 3182 D8        START  CLD             ;Entry, initialize
300 3183 20E131           JSR SWAP        ;Store zero page locations
310 3186 A900             LDA #0
320 3188 8500             STA CNTL        ;Reset counter to zero
330 318A 8501             STA CNTH
340 318C 8504             STA LAST        ;Reset signal flag to zero
350 318E A920             LDA #MUXSET     ;Set multiplexer control
360 3190 8D0AC7           STA MUX
370 3193 AD7E31           LDA DR          ;Set duration counter
380 3196 8502             STA DRCL
390 3198 AD7F31           LDA DR+1
400 319B 8503             STA DRCH
410 319D AD08C7    FREQ6  LDA INPUT       ;Input from A/D
420 31A0 3030             BMI FREQ1       ;Check for positive
430 31A2 A504             LDA LAST
440 31A4 D030             BNE FREQ2       ;Check for transition
450 31A6 E600             INC CNTL        ;On transition
460 31A8 D004             BNE FREQ3       ;  increment counter
470 31AA E601             INC CNTH
480 31AC D005             BNE FREQ8
490 31AE EA        FREQ3  NOP             ;Delay 7 machine cycles
500 31AF A901             LDA #01
510 31B1 D000             BNE FREQ8
520 31B3 A901      FREQ8  LDA #01         ;Set signal flag (LOW)
530 31B5 8504             STA LAST
540 31B7 C602      FREQ5  DEC DRCL        ;Decrease duration count
550 31B9 D012             BNE FREQ7
560 31BB C603             DEC DRCH
570 31BD 10DE             BPL FREQ6       ;Check for zero
580 31BF A500             LDA CNTL        ;  duration remaining
590 31C1 8D8031           STA RES         ;At end, place frequency
600 31C4 A501             LDA CNTH        ;  count into RES
610 31C6 8D8131           STA RES+1
620 31C9 20E131           JSR SWAP        ;Restore zero page
630 31CC 60               RTS             ;And return
640 31CD EA        FREQ7  NOP             ;Delay 8 machine cycles
650 31CE A900             LDA #0
660 31D0 F0C8             BEQ FREQ6
670 31D2 A900      FREQ1  LDA #0          ;6 cycles including jump
680 31D4 8504             STA LAST        ;Reset Signal Flag (HIGH)
690 31D6 EA        FREQ2  NOP             ;Delay 25 machine cycles
700 31D7 EA               NOP
710 31D8 EA               NOP
720 31D9 EA               NOP
730 31DA EA               NOP
740 31DB EA               NOP
750 31DC EA               NOP
760 31DD A900             LDA #0
770 31DF F0D6             BEQ FREQ5       ;Then decrement duration
780                ;
790                ;  Subroutine to store zero page locations
800                ;    in STORE and to restore zero page
810                ;    prior to return.
820                ;
830 31E1 A204      SWAP   LDX #04
840 31E3 B500      NEXT   LDA $0,X
850 31E5 A8               TAY
860 31E6 BDF331           LDA STORE,X
870 31E9 9500             STA $0,X
880 31EB 98               TYA
890 31EC 9DF331           STA STORE,X
900 31EF CA               DEX
910 31F0 10F1             BPL NEXT
920 31F2 60               RTS
930                ;
940                ;
950 31F3 00        STORE .BYTE 0,0,0,0,0 ;Zero page storage location
950 31F4 00
950 31F5 00
950 31F6 00
950 31F7 00
```

The values for the capacitor and the resistor should he kept large, but are not critical. The capacitor should be non-polarized.

The variable LAST is used to store the input state previously encountered. If a negative value is input, the variable LAST is set to 0. When the input is positive, LAST is checked. A value of LAST equal to zero indicates that a positive transition has occurred and the frequency count is incremented. If the input is not positive or if LAST is positive, the program enters a delay loop to equalize loop timing.

The Program

A listing of the assemhly language program is presented in listing 1. Five zero page locations ($00-$04) are used for the flag LAST, and for counting, and to minimize the number of machine cycles in the longest path through the loop. Since this program was intended to be called as a USR(X) subroutine from a BASIC driver, the contents of the zero page locations used are saved in a temporary storage location hy the subroutine SWAP (line 830). The input from the A/D converter is at C708H. This variable, called INPUT, will have to be changed to suit the user's specific equipment. Similarly, MUX is the address of an analog multiplexer, set to select port "0" in this program.

The assembly program is written with a starting location memory address of 317E H. This location was selected to correspond to the beginning of available work space for the Ohio Scientific OS65D V3.2 disk operating system provided with 8-inch floppy disk systems. The BASIC driver can be placed after the machine code in the workspace, which permits storage and recall of the object code and the BASIC driver in a single named disk file. Details of how this is accomplished are covered in the OSI disk operating system manual and will not be covered here.

However, the starting location for the assembly is not important; the machine code can be moved to any available memory locations that can be protected to prevent interference by the BASIC program. The execution cycles for some 6502 instructions used in the program depend on whether or not references are made across page boundaries. The program timing has been computed assuming that all branches occur within a single page. Therefore, if the program is relocated, be careful to place the entire program within a page of memory.

Table 1

	Approximate Voltage Level	8-Bit Binary Representation	Decimal Equivalent
+ Full Scale	+ 10 Volts	00000000	0
Mid Scale	0 Volts	01111111	127
– Full Scale	– 10 Volts	11111111	255

Listing 2

```
100 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
105 PRINTTAB(15);"****************************"
110 PRINTTAB(15);"*                          *"
115 PRINTTAB(15);"*    FREQUENCY  COUNTER    *"
120 PRINTTAB(15);"*  Program by P Lindquist  *"
125 PRINTTAB(15);"*    Union Lake, Mich.     *"
130 PRINTTAB(15);"*       January 1980       *"
135 PRINTTAB(15);"*                          *"
140 PRINTTAB(15);"****************************"
145 PRINT
150 REM  SET USR SUBROUTINE ADDRESS TO $3182
155 POKE 8955,130 : POKE 8956,49
160 S=0 : S2=0 : V=0
165 FORI=1TO10 : PRINT : NEXT
170 PRINT"    Options available:"
175 PRINT"       1)  Count"
180 PRINT"       2)  Calibrate"
185 PRINT"       3)  Return to monitor"
190 PRINT
195 INPUT"    Option desired (1-3)";N
200 PRINT
205 N = INT(N+.5)
210 IF N=3 THEN RUN"BEXEC*
215 IF N=2 THEN 340
220 IF N=1 THEN 230
225 GOTO 170
230 PRINT"*** COUNT ***"
235 PRINT
240 INPUT"    Input desired number of frequency samples";N
245 PRINT
250 IF N<1 THEN 170
255 PRINT : PRINT : PRINT : PRINT : PRINT
260 PRINT "  SAMPLE","FREQUENCY","  MEAN","STANDARD"
265 PRINT "  NUMBER",TAB(42);"DEVIATION"
270 PRINT
275 REM  FOR A DISCUSSION OF STATISTICAL APPROXIMATIONS SEE
280 REM  JAN 79 BYTE ARTICLE BY A B FORSYTHE.
285 FOR I = 1 TO N
290 X=USR(X)
295 X=256*PEEK(12673)+PEEK(12672)
300 D=X-S
305 S=S+D/I
310 S2=S2+D*(X-S)
315 IF I>1 THEN V=S2/(I-1)
320 SD=SQR(V)
325 PRINTTAB(3);I,X,S,SD
330 NEXT I
335 GOTO 160
340 PRINT"*** CALIBRATE ***"
345 PRINT
350 N = PEEK(12670) + 256*PEEK(12671)
355 PRINT"    Currently the program is set to count for ";N
360 PRINT"    cycles through the machine code counter routine."
365 PRINT"    Each count cycle requires 46 microprocessor cycles.
370 PRINT
375 INPUT"    Would you like to change this";A$
380 PRINT
385 IF LEFT$(A$,1)<>"Y" THEN 160
390 INPUT"    Input new number of count cycles";N
395 PRINT
400 IF N<1 OR N>32767 THEN PRINT"*** RANGE PROBLEM ***":GOTO160
405 S=INT(N/256) : POKE 12671,S : POKE 12670,N-256*S
410 INPUT"    Would you like to update the disk file";A$
415 PRINT
420 IF LEFT$(A$,1)<>"Y" THEN 160
425 DISK!"PU FRQCNT
430 GOTO 160
```

A pair of 2-byte variables is provided for communication with a driver routine. DR is an input to the program and determines the number of cycles through the count loop. This number must be less than 32767. RES is the return location for the frequency count. These locations may be accessed from the BASIC driver using PEEK and POKE statements.

The NOP's beginning in line 690 provide the delay necessary to balance the number of machine cycles in the two major program loops. There are, in addition, two shorter delay loops in program lines 490 and 640 which are entered when it is not necessary to increment the frequency count high byte or decrement the duration counter high byte, respectively.

The count program is written as a subroutine and is terminated by a RTS (return from subroutine). Zero page locations are restored, but the microprocessor registers are not saved.

The Driver

Listing 2 is a Microsoft BASIC driver program. Remember that the machine code counter is loaded with the BASIC driver by the OSI OS65D3 operating system, so that an independent load of the machine code by the BASIC program is not required. Loading of the machine code by the BASIC driver may be required by other operating systems, however.

The driver has three major sections: Initialization and menu (lines 150-225), count (lines 230-335), and calibrate (lines 340-430). Initialization consists of first POKEing the starting address 3182H (decimal 12674) into the memory locations that control the jump to subroutine from the BASIC USR(X) command (locations 8955 and 8956 for OSI'S OS65D3 system). Secondly, it initializes variables S, S2, and V, which are used for statistical analysis. The menu requests the user to select the count function (1), the calibrate function (2), or stop (3).

The count function will call (in line 290) the machine code frequency counter the number of times specified by the user. The frequency count is returned in memory locations 12672 (low byte) and 12673 (high byte) with the machine code assembled with a starting location of 12670 (317EH). For each sample, the sample number, the actual frequency count, the mean frequency count, and the standard deviation are printed. The statistical

Figure 2

```
         ****************************
         *                          *
         *    FREQUENCY  COUNTER    *
         *  Program by P Lindquist  *
         *    Union lake,  Mich.    *
         *       January 1980       *
         *                          *
         ****************************


Options available:
  1)  Count
  2)  Calibrate
  3)  Return to monitor

Option desired (1-3)? 1

COUNT ***

input desired number of frequency samples? 8



SAMPLE    FREQUENCY    MEAN           STANDARD
NUMBER                                DEVIATION

  1         4807       4607           0
  2         4808       4807.5         .707106781
  3         4808       4807.66667     .577350407
  4         4808       4807.75        .500000159
  5         4806       4807.8         .447213782
  6         4808       4807.83333     .408248548
  7         4808       4807.85714     .377964795
  8         4808       4807.875       .353553775
```

```
Option desired (1-3)? 2

CALIBRATE ***

Currently the program is set to count for  21739
cycles through the machine code counter routine.
Each count cycle requires 46 microprocessor
cycles.
Would you like to change this? Y

Input new number of count cycles? 21709

Would you like to update the disk file? N

Option desired (1-3)? 1

COUNT ***

Input desired number of frequency samples? 12

SAMPLE    FREQUENCY    MEAN          STANDARD
NUMBER                               DEVIATION

  1         4801       4801          0
  2         4801       4801          0
  3         4801       4801          0
  4         4802       4801.25       .5
  5         4802       4801.4        .547722623
  6         4802       4801.5        .547722645
  7         4802       4801.57143    .53452258
  8         4801       4801.5        .534522548
  9         4801       4801.44445    .527046346
 10         4801       4801.4        .516397833
 11         4801       4801.36364    .504525001
 12         4801       4801.33334    .492365995
```

algorithms used are discussed in an article by A.B. Forsythe in the January 1979 issue of *Byte* magazine.

If the microprocessor clock speed is 1 MHz, 21739 cycles through the count routine will require one second. The frequency count returned will be in cycles per second or Hz. If the microprocessor clock speed is not a nominal 1 MHz, the loop count may be adjusted in the CALIBRATE section of the drive program. The count value to be adjusted is contained in memory locations 12670 (low byte) and 12671 (high byte). Line 425 resaves the program with the new frequency count utilizing the OS65D3 disk operating system command !" PUT FILENAME. The author's system, adjusted with a calibrated frequency generator, is set for a loop count of 21705.

The Results

Figure 2 demonstrates the statistical output provided by the BASIC driver routine. The frequency source used in this example was the 4800 Hz provided to an ACIA which, when divided by 16 results in 300 baud to a modem. The program has been useful on several occasions for measuring frequencies between 60 and 10,000 Hz. Success with this program has inspired other programs to simulate electronic test equipment. Audio oscillators and audible logic probes (using the D to A output on the CA-22 board) are relatively straight forward. An oscilloscope simulator has also been developed, but its frequency response is limited by program speed. In each case the technique of combining machine code with BASIC drivers, as is permitted using the OSI disk operating system, has proven very beneficial. This technique allows a balance in program speed, programming ease, program flexibility, availability of high level mathematical functions, and display flexibility.

References

1. "Microprocessor-Based Analog/ Digital Conversion", R. Frank, *Byte*, May 1976, page 70.

2. "Interfacing With An Analog World, Part 2", J. Carr, *Byte*, June 1977, page 54.

3. "Elements Of Statistical Computation", A. Forsythe, *Byte*, January 1979, page 182.

Note: Information concerning the CA-22 Analog to Digital Converter Module was provided courtesy of Ohio Scientific Inc.

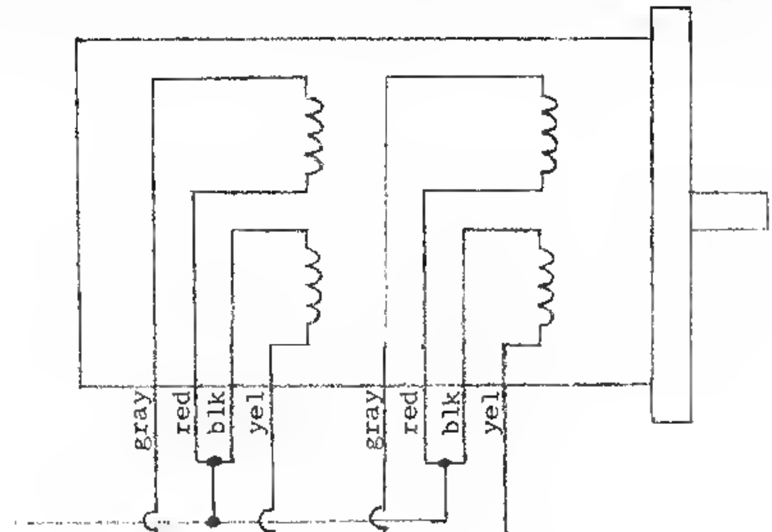
MICRO

Step Up to Programmable Motion

Making objects move at the command of your computer is a simple and inexpensive proposition. This article describes the interfacing of a stepper motor to a KIM-1. The applicability is demonstrated in a programmable flow metering system.

David S. Liscinsky
Old Colony Lane
Cromwell, CT 06416

Figure 1: **A typical unipolar stepper motor schematic and switching sequence.**



Q1	Q2	Q3	Q4	STEP
ON	OFF	ON	OFF	1
ON	OFF	OFF	ON	2
OFF	ON	OFF	ON	3
OFF	ON	ON	OFF	4
ON	OFF	ON	OFF	5

← clockwise (down) — counter-clockwise (up) →

Many exciting computer applications require the ability to physically move objects. These applications particularly impress those who continually ask, "but what can it do?", referring to the microcomputer system you've been working with for months. One simple solution to programmable motion is a stepper motor interfaced to an I/O port of your computer. This approach is not only inexpensive, but allows the very precise movements necessary for many applications, such as: an x-y plotter, moveable "arms" for robots and interactive game playing, and the simple opening and closing of valves. This article will discuss how to go about hooking up a stepper motor(s) to your computer. Then an actual working system configured around a KIM-1 will be used as an example of how to cause movement at the command of your microcomputer.

Because stepper motors convert electrical pulses into discrete mechanical movements, it is easier to obtain very precise motion from a stepper motor than from any other type. Each move of the motor is controlled by switching the current to its windings on and off in the proper sequence. Figure 1 is a diagram of a typical stepper motor with four separate windings and a table of the appropriate switching sequence. Energizing the windings with each patten in the sequence rotates (steps) the motor shaft a predetermined angle from 3.75° to 90°. If bits from an output port of a microcomputer are used to control the current switching, the motor shaft can easily be rotated any number of steps in either direction by software commands. For an excellent discussion on the details of how stepper motors actually work, see reference 1.

Selection of the proper stepper motor can be difficult. However, for most applications, consideration of torque and step angle is all that is necessary. The amount of torque required is determined by the application load, that is, how much energy it will take to budge the object you want to move. The torque developed by a specific motor is a function of how fast you try to turn it, the amount of current that is supplied to the windings, and the drive design. The speed/torque curves supplied by the manufacturers to describe their motors can aid in proper matching of a motor to your application. However, torque is not a function of cost. Therefore, it simplifies matters and is cost-effective

to get a motor that can supply more torque than you actually need.

The second consideration is step angle, or how precisely the rotation can be controlled. Typically the choice is 7.5° (48 steps/rev). In general, smaller step angles cost significantly more, but larger angles do not. So again, to simplify matters, the smallest step angle for the money is usually the best choice since it can be used in more applications. Applications that require more resolution than 7.5°/step can use gearing.

Figure 2 is an inexpensive drive interface between a parallel I/O port and a stepper motor with 4-coil unipolar windings. By writing the appropriate bit patterns to the port in the proper sequence, the motor shaft can be rotated clockwise or counter-clockwise. Listing 1 is a simple program to rotate the shaft clockwise, one revolution, using the two least significant bits of port B from the on-board 6530 of a KIM-1.

A Real Application

A particularly simple but quite useful application for a stepper motor is the programmable opening and closing of valves to control fluid flow. Although this application is probably of more interest to industrial process control than microcomputer hobbyists, it illustrates the simplicity of using stepper motors and software to control motion.

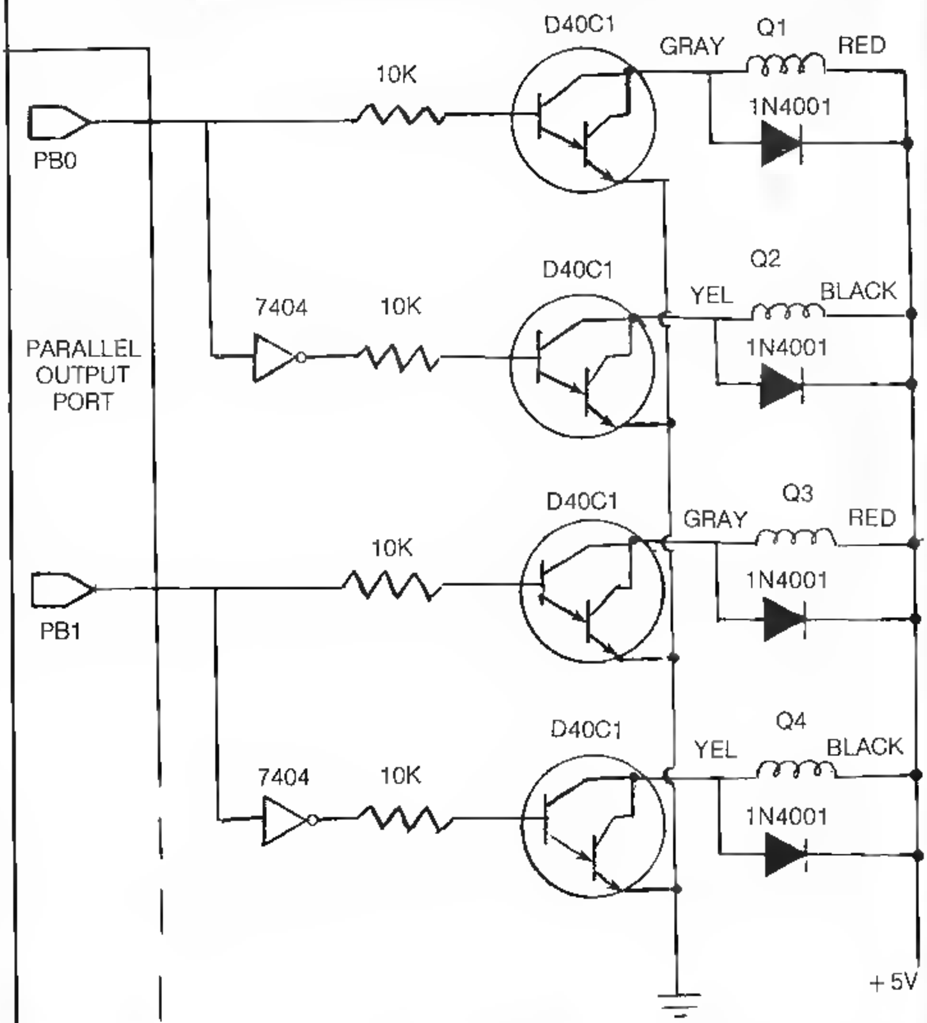
The Hardware

The first problem in controlling fluid flow is the mechanical coupling of a valve and the stepper motor shaft. One possible coupling is shown in photo 1. Other approaches would include "gear"-type drive systems. However, direct coupling is the simplest method.

The next problem is in knowing the flow at any given position of the valve. Ideally, flow control would be closed-loop. That is, a transducer would feed back a signal that is proportional to the flow to the computer. The executing program would then modify the flow by opening or closing the valve in order to maintain the desired flow. However, flow transducers are expensive and dependent on the type of fluid then are sensing.

An alternate technique is the execution of a preselected sequence of instructions, or open-loop control. Although this is a less flexible means of control, it is also less expensive. All that is required is calibration of the valve position *vs.* flow, or more specifically, steps

Figura 2: **Inexpensive drive interface for e stapper motor with 4 coil unipolar windings.**



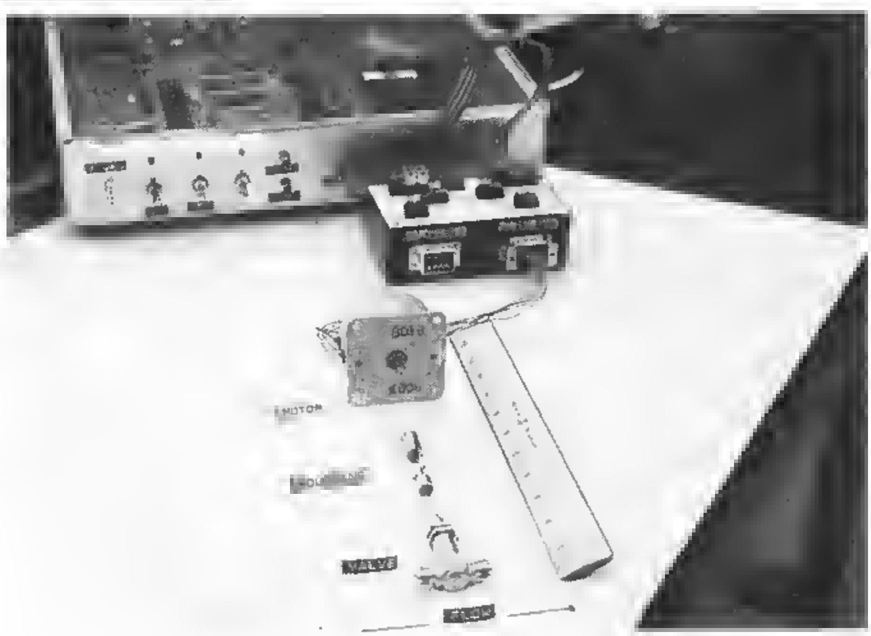


Photo 1: **The mechanicel coupling of e stepper motor and a metaring valve.**

of the motor *vs.* flow. These values can then be used as settings for various flow rates allowing almost any flow *vs.* time pattern to be generated.

The Software

Program development and user interaction (to produce complex flow patterns) is rather easy on a relatively powerful computer such as an AIM 65. However, an inexpensive, yet sophisticated, open-loop flow control system can be based on a single board computer like the KIM-1. The software presented here reflects a limited amount of computing power, and depends on the user precalculating and entering the command parameters. The driving program (listing 2) interprets and executes the entered table of instructions. The program is easy to use and illustrates the control over motion that can be inexpensively obtained.

The objective of the program is to allow almost any flow *vs.* time pattern to be generated. The user enters values in a table that can occupy up to 255 contiguous bytes in memory. The table is composed of up to 51 unique move commands (5 bytes each) that specify successive flow settings. Increasing the complexity of the desired curve increases the total number of commands that are required.

Table 1 defines the move command. In general, the arguments of the command specify:

1. how long to hold a given flow,
2. how many steps to rotate at the end of the hold time,
3. how many times to repeat a particular sequence of commands,
4. the starting address of a sequence to be repeated.

All entries are in hexadecimal. Loops may not be nested, but otherwise, anything goes.

Listing 1 also includes a sample command table which will open (⅓ rev.) and close (⅓ rev.) the valve 16 times at 5-second intervals, followed by a 10-second wait and two clockwise revolutions.

Conclusion

The interfacing of a stepper motor to a KIM-1 has been described. The concepts were implemented in a programmable flowmetering system. Throughout the discussion, low cost

***Table 1:* Definition of a Move Command**

Address	Description
XXX0	HOLD time in minutes
XXX1	HOLD time in seconds
XXX2	Number of steps to ROTATE a. 00 = no steps b. 01 to 7F = number of CW steps c. 80 to FF = number of CCW steps + $7F_H$
XXX3	Number of times to repeat (LOOP) a sequence a. 00 = no loop, go directly to the next HOLD time b. 01 to FE = number of times to loop c. FF = stop
XXX4	BRANCH ADDRESS a. 00 to FF = low order byte of the address of the next HOLD time

***Listing 1:* A KIM-1 program to rotate a stepper motor one clockwise revolution in 10 seconds.**

```
                 ;* FILE ROTCW
                 ;*
                 DDR    EQU $1703
                 PORT   EQU $1702
                 ;
                        ORG $2D0
                        OBJ $800
                 ;
0200 A9FF               LDA #$FF         ;INITIALIZE PORT B FOR OUTPUT
0202 8D0317             STA DDR
0205 D8                 CLD
0206 A900               LDA #$00
0208 8D3602             STA STEPS        ;INITIALIZE STEP COUNT
020B A203        LP1    LDX #$03
020D BD3202      LP2    LDA TABL,X       ;GET APPROPRIATE BIT PATTERN
0210 8D0217             STA PORT         ;ROTATE 1 STEP
0213 A9CB               LDA #$CB
0215 8D0717             STA $1707        ;DELAY 20.8 MSEC.
0218 AD0617      LP3    LDA $1706
021B D0FB               BNE LP3
021D AD3602             LDA STEPS
0220 18                 CLC
0221 6901               ADC #$01
0223 C930               CMP #$30         ;IF COUNT=48,
                                          THEN 1 REV. COMPLETED
0225 F008               BEQ HALT
0227 8D3602             STA STEPS        ;OTHERWISE,
                                          SET X FOR NEXT PATTERN
022A CA                 DEX
022B 10E0               BPL LP2
022D 30DC               BMI LP1
022F 20051C      HALT   JSR $1C05
0232             ;
0232 020001      TABL   HEX 02000103
0235 03
0236             ;
0236 00          STEPS  BYT $00
                        END
```

and simplicity have been emphasized. However, the same concepts can be used for complex control situations in which many simultaneous movements are desired. This discussion is just the first step.

References

1. Bober, R.E. "Taking the First Step," *BYTE*, Feb. 1979, p. 35-112.
2. Margolin, Bob. "Stepping Motors," *Electronics Products Magazine*, Sept. 1979, p. 35-44.
3. Hill, John W. "Introducing the Mini-Mover 5," *Robotics Age*, Summer, 1980, p. 18-27.
4. *Stepper Motor Handbook*. North American Philips Controls Corp., Cheshire Industrial Park, Cheshire, CT 06410.

David Liscinsky is an Associate Research Scientist at United Technologies Research Center in East Hartford. His interests include real-time processing and optimization of experiments using computer control.

Listing 2: KIM-1 program to provide open-loop control of a one valve flowmetering system.

```
                     ;* OPEN LOOP CONTROL
                     ;*  PROGRAM FOR KIM
                     ;* BY DAVID LISCINSKY
0200 202002    MAIN   JSR INIT        ;INITIALIZE
0203 205302    NEXT   JSR TIME        ;HOLD FOR REQUESTED TIME
0206 207602           JSR MOVE        ;ROTATE REQUESTED NO. STEPS
0209 BD0003           LDA TABL,X      ;REPEAT A SEQUENCE?
020C F00A             BEQ CONT        ;NO
020E C9FF             CMP #$FF
0210 F00B             BEQ STOP
0212 202E02           JSR LOOP
0215 AEF102           LDX TX
0218 E8        CONT   INX
0219 E8               INX
021A 4C0302           JMP NEXT
021D 20051C    STOP   JSR $1C05
0220           ;
0220 A9FF      INIT   LDA #$FF        ;INITIALIZATION SUBROUTINE
0222 8D0317           STA $1703       ;PORT B DATA DIRECTION
0225 A200             LDX #$00
0227 8EEE02           STX PTR         ;OTHER POINTERS AND COUNTERS
022A 8EEF02           STX KNT
022D 60               RTS
022E           ;
022E 8DEF02    LOOP   STA KNT         ;COUNT FOR LOOP
0231 204A02    L10    JSR GOTO
0234 205302    L11    JSR TIME
0237 207602           JSR MOVE
023A BD0003           LDA TABL,X
023D F006             BEQ NX
023F CEEF02           DEC KNT
0242 D0ED             BNE L10
0244 60               RTS
0245           ;
0245 E8        NX     INX
0246 E8               INX
0247 4C3402           JMP L11
024A           ;
024A 8EF102    GOTO   STX TX          ;TRANSFER THE BRANCH ADDRESS
024D E8               INX             ; TO THE X REGISTER, THEREBY
024E BD0003           LDA TABL,X      ; RESETTING THE INDEX OF THE
0251 AA               TAX             ;THE NEXT TABLE LOCATION TO
0252 60               RTS             ; BE INTERPRETED
0253           ;
0253 BD0003    TIME   LDA TABL,X      ;GET NO. OF MINUTES TO WAIT
0256 F00B             BEQ PS1
0258 8DEC02           STA VAR
025B A93C             LDA #$3C
025D 8DF002           STA TEMP
0260 209102           JSR DELY        ;DELAY FOR 60 SECONDS
0263 E8        PS1    INX
0264 BD0003           LDA TABL,X      ;GET NO. OF SECONDS TO WAIT
0267 F00B             BEQ PS2
0269 8DEC02           STA VAR
026C A901             LDA #$01
026E 8DF002           STA TEMP
0271 209102           JSR DELY        ;DELAY FOR 1 SECOND
0274 E8        PS2    INX
0275 60               RTS
0276           ;
0276 BD0003    MOVE   LDA TABL,X
0279 F00A             BEQ RTRN
027B 8DEC02           STA VAR         ;VAR= TOTAL # STEPS TO TAKE
027E C980             CMP #$80
0280 B005             BCS BW          ;CCW IF >=$80
0282 20B102           JSR CW          ;OTHERWISE CW
0285 E8        RTRN   INX
0286 60               RTS
0287 4980      BW     EOR #$80
0289 8DEC02           STA VAR
028C 20C702           JSR CCW
028F E8               INX
0290 60               RTS
0291           ;
0291 ADF002    DELY   LDA TEMP        ;USES "DIVIDE BY 1024"
0294 8DED02           STA VAR2        ; INTERVAL TIMER TO DELAY
0297 A004      L2     LDY #$04        ; FOR NO. OF SECONDS IN
```

(Continued)

```
0299 A9F4      L3     LDA #$F4          ; ACCUMULATOR
029B 800717           STA $1707
029E AD0617    L4     LDA $1706
02A1 D0FB             BNE L4
02A3 88               DEY
02A4 D0F3             BNE L3
02A6 CEED02           DEC VAR2
02A9 D0EC             BNE L2
02AB CEEC02           DEC VAR
02AE D0E1             BNE DELY
02B0 60               RTS
02B1           ;
02B1 ACEE02    CW     LDY PTR
02B4 20DB02    L5     JSR STEP          ;TAKE 1 STEP CW
02B7 C8               INY
02B8 C004             CPY #$04          ;AFTER 4 STEPS RESET Y REG. FOR
02BA D002             BNE L6            ; BEGINNING OF STEPPING SEQUENCE
02BC A000             LDY #$00
02BE CEEC02    L6     DEC VAR
02C1 D0F1             BNE L5            ;IF VAR=0, ROTATION IS FINISHED
02C3 8CEE02           STY PTR
02C6 60               RTS
02C7           ;
02C7 ACEE02    CCW    LDY PTR
02CA 88        L7     DEY
02CB 1002             BPL L8
02CD A003             LDY #$03          ;AFTER 4 STEPS RESET Y REG. FOR
02CF 20DB02    L8     JSR STEP          ; START OF STEPPING SEQUENCE
02D2 CEEC02           DEC VAR           ;TAKE 1 STEP CCW
02D5 D0F3             BNE L7
02D7 8CEE02           STY PTR
02DA 60               RTS
02DB           ;
02DB B9F202    STEP   LDA TURN,Y        ;USE Y REG. TO INDEX TURN TABLE
02DE 8D0217           STA $1702         ;TAKE 1 STEP
02E1 A90A             LDA #$0A
02E3 8D0717           STA $1707         ;DELAY ABOUT 0.01 SEC. TO
02E6 AD0617    L9     LDA $1706         ; ALLOW COILS TO RELAX
02E9 D0FB             BNE L9
02EB 60               RTS
02EC 00        VAR    BYT $00           ;TOTAL NO. OF STEPS TO TAKE
02ED 00        VAR2   BYT $00           ;SCRATCH
02EE 00        PTR    BYT $00           ;POINTER TO NEXT PATTERN IN SEQ.
02EF 00        KNT    BYT $00           ;NO. TIMES TO REPEAT A LOOP
02F0 00        TEMP   BYT $00           ;SCRATCH
02F1 00        TX     BYT $00           ;SCRATCH
02F2           ;
02F2           ;TURN: TABLE OF BIT PATTERNS TO PROVIDE
02F2           ;      THE APPROPRIATE STEPPING SEQUENCE
02F2           ;      FOR A 4-WINDING UNIPOLAR STEPPER MOTOR
02F2           ;
02F2 030100    TURN   HEX 03010002
02F5 02
02F6           ;
0300                  ORG $300
0300           ;
0300 00        TABL   BYT $00           ;SAMPLE: HOLD 0 MIN.
0301 05               BYT $05           ;             5 SEC.
0302 10               BYT $10           ;        ROTATE 16 STEPS CW
                                                 (1/3 REV.)
0303 00               BYT $00           ;GO TO NEXT COMMAND
0304 00               BYT $00
0305 00               BYT $00           ;        HOLD 0 MIN.
0306 05               BYT $05           ;             5 SEC.
0307 90               BYT $90           ;        ROTATE 16 STEPS CCW
0308 10               BYT $10           ;LOOP 16 TIMES
0309 00               BYT $00           ; STARTING AT TABLE ADDR. $XX00
030A 00               BYT $00           ;        HOLD 0 MIN.
030B 0A               BYT $0A           ;             10 SEC.
030C 60               BYT $60           ;        ROTATE 96 STEPS
                                                 (2 REVS.)
030D FF               BYT $FF           ;STOP
030E           ;
                      END
```


MICRO

MICRO

New Publications

Mike Rowe
New Publications
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Software

Graphic Software for Microcomputers by B.J. Korites. Kern Publications (190 Duck Hill Road, P.O. Box 1029, Duxbury, MA 02332), 1981, 184 numbered pages, illustrations, listings, 11 × 8½ inches, cardstock cover with plastic comb binding. $19.95

A self-teaching guide to writing computer graphics software on microcomputers. It contains 61 programs for 2 and 3 dimensional graphics ranging from elementary to advanced concepts. Theoretical concepts are presented next to actual program listings so that users can modify programs to suit their own applications. Because the book has been printed on only one side of each leaf, there are actually 368 pages, 184 of them blank. Students can use the blank page facing each text page for notes. All programs are in BASIC, written for the Apple II Plus 48K system and may be obtained on disk from the publisher for $18.95. The author states that the programs are easily convertible to other languages, such as FORTRAN or Pascal.

CONTENTS: *Introduction; Basic Plotting Commands; Point Drawings; Line Drawings; 2D-Interactive Graphics; 2D-Translation; 2D-Scaling and Stretching; 2D-Clipping; 2D-Rotation; 3D-Rotation; 3D-Translation and Rotation; Perspective; Intersections; Hidden Line Removal; Shading; 3D-Shapes; Matrix Concatenations; Tablets; Applications; Practice Problems.*

Nailing Jelly to a Tree by Jerry Willis and William Danley, Jr. dilithium Press (P.O. Box 606, Beaverton, OR 97075), 1981, viii, 244 pages, diagrams, listings, 5½ × 8½ inches, paperbound.
ISBN: 0-918398-42-8 $12.95

This is a book on software for the person who is interested in using and adapting the many computer programs available in books and magazines and from software houses.

CONTENTS: *Introduction to the Care and Feeding of Small Computers*—Who is this book for?; What you will read about; Levels of computer language. *Two, Four, Six, Eight—What You Gonna Accumulate!*—The binary number system; Binary math; Types of numbers; Boolean algebra; Computer codes. *Software I Have Known and Loved*—Starting options; Some typical monitor and operating systems software. *Mr. Chips and the Machine Language*—Chips and instruction sets; Registers and flags; Instruction sets; Hardware details and software operation; Important memory addresses; A sample machine language program. *A Better Way—Assembly Language Programming*—Steps in assembly language programming. *Quick and Dirty BASIC*—Getting acquainted; How to use this chapter; Introduction to BASIC. *More Dirt*—Making decisions and comparisons; Subscripted variables; Math functions; String functions. *Converting from One BASIC to Another*—Step 1. Is it possible?; Step 2. Make clerical changes; Step 3. Content changes. *A Basic Glossary and Conversion Guide*—Symbols and punctuation marks. *Appendix Conversion Table. Index.*

Educational Software Directory — Apple II Edition by Sterling Swift Publishing Co. (P.O. Box 188, Manchaca, Texas 78652), 1981, viii, 104 pages, 5½ × 8½ inches, plastic comb binding with cardstock cover.
ISBN: 0-88408-141-9 $11.95
(Education price, $9.95)

This directory lists hundreds of software packages available from 58 commercial publishers of educational software and from 7 noncommercial publishers. The publishers of the directory have listed programs in these areas: Computer Literacy (programs teaching BASIC or having anything to do with the Apple); Computer-Assisted Instruction (CAI) or programs which teach or instruct in any way (tutorial, drill and practice, simulation, problem solving, teaching aids, games); Administrative (teacher-grading programs, school registration programs, etc.); and Statistical Packages (for calculating mean, standard deviation, etc.).

Special annotation indicates if a program was designed for classroom use; represents a large number of hours of curriculum material; uses Applesoft or Integer BASIC; is available on disk or cassette; and the amount of memory required.

Programs are listed by publisher, with prices usually given. A comprehensive index lists programs by title under four school-level headings: Elementary; Middle School (Junior High); Secondary; and Community College, College/University, Continuing Education. A separate index heading, overlooked in preparing the Table of Contents, lists titles of administrative software, courseware development, and utility software. The directory contains no advertising.

Fifty BASIC Exercises by Jean-Pierre Lamoitier. Sybex, Inc. (2344 Sixth Street, Berkeley, California 94710), 1981, xx, 232 pages, approximately 200 figures (charts, diagrams, listings), 7 × 9 inches, paperbound.
ISBN: 0-89588-056-3 $12.95

This tutorial is designed to teach BASIC through graduated exercises. It is written for persons with a minimum scientific or technical background. The programs in the book are written in Microsoft BASIC, which, the author says, "will execute directly on a TRS-80, and with occasional small changes, on a PET/CBM, APPLE, or any other popular computer equipped with Microsoft BASIC." Each exercise includes a statement of the problem to be solved, an analysis of the problem, a solution with flowchart and comments, the corresponding program, and a sample run.

CONTENTS: *Your First Program in BASIC*—Introduction; Computing Taxable Income; Another Way to Calculate Taxable Income; Conclusion. *Flowcharts*—Introduction; The Purpose of the Flowchart; The Maximum of Two Numbers, A and B; Example of a Complete Flowchart: The Largest Element of an Array; How to Verify a Flowchart; Decision Points; A "Flip-Flop" Technique for Branching; The Implementation of a P-stage Round Robin; Conclusion. *Exercises Using Integers*—Introduction; Integers Satisfying $A^2 + B^2 = C^2$; Armstrong Numbers; Partitioning a Fraction into Egyptian Fractions; Prime Numbers; Decomposition into Prime Factors; Conversion from Base Ten to Another Base; Conclusion. *Elementary Exercises in Geometry*—Introduction; The Area and Perimeter of a Triangle; Determination of a Circle Passing Through Three Given Points; Computing the Length of a Fence; Plotting a Curve; Conclusion. *Exercises Involving Data Processing*—Introduction; Shell Sort; Merging Two Arrays; The Day of the Week; The Time Elapsed Between Two Dates; A Telephone Directory; Conclusion. *Mathematical Computations*—Introduction; Synthetic Devision of a Polynomial by (X - S); The Calculation of a Definite Integral; Calculation of π Using Regular Polygons; Solving an Equation by Dichotomy; Numerical Evaluation of Polynomials; Conclusion. *Financial Computations*—Introduction; Sales Forecasting; Repayment of Loans; Calculation of the Rate of Growth; More on Income Taxes; The Effect of Additional Income on Purchasing Power; Conclusion. *Games*—Introduction; The Game: TOO LOW/TOO HIGH; Finding an Unknown Number by Bracketing; The Matchstick Game; The Game of Craps;

(Continued on next page)

New Publications

(Continued from page 35)

Conclusion. *Operations Research*—Introduction; Topological Sort; The Critical Path in a Graph; The Traveling Salesman Problem; Conclusion. *Statistics*—Introduction; The Average of a Sequence of Measurements; Mean, Variance and Standard Deviation; Linear Regression; The Distribution of Random Numbers Obtained from the Function RND; Conclusion. *Miscellaneous*—Introduction; The Signs of the Zodiac; The Eight Queens Problem; Conclusion. *Appendices*—*A.* The Alphabet in BASIC; *B.* Main Syntax Rules. *Index.*

PET/CBM

PET/CBM Personal Computer Guide, Second Edition, by Adam Osborne and Carroll S. Donahue. OSBORNE/McGraw-Hill (630 Bancroft Way, Berkeley, California 94710), 1980, x, 500 pages, photographs, drawings, listings, 6½ × 9 3/16 inches, paperbound. ISBN: 0-931988-55-1 $15.00

This edition is a major revision of the original book of the same title, also published in 1980, but authored by Carroll S. Donahue and Janice K. Enger. Adam Osborne has transformed the book into a BASIC and CBM BASIC tutorial. The book describes all models of CBM computers as of publication date, as well as software products introduced by Commodore.

CONTENTS: *Introducing CBM Computers*—CBM Features. *Operating the CBM Computer*—Intermediate Mode; Program Mode; Operating the Cassette Units; Operating the Disk Unit; Operating the CBM Printer. *Screen Editing. Programming the CBM Computer*—Elements of a Programming Language; BASIC Statements; Functions. *Making the Most of CBM Features*—Hardware Features; String Concatenation; Input and Output Programming; Mathematical Programming; Graphics; The Real Time Clock; Random Numbers. *Peripheral Devices: Tape Cassette Drives, Diskette Drives and the Printer*—Storing Data on Magnetic Surfaces; Cassette Files; Diskette Files; Diskette Housekeeping Operations; Sequential Data Files; Relative Data Files (BASIC 4.0); Using GET# with Diskette Files; Program Files; Programming the Line Printer. *System Information. CBM BASIC*—BASIC Statements; Functions; CBM 8000 Editing Functions. *Appendices*—*A.* CBM Character Codes; *B.* CBM Error Messages; *C.* BASIC Bibliography; *D.* CBM Newsletters and References; *E.* Conversion Tables; *F.* Revision Level 2 ROMs. *Index.*

(Continued on page 47)

Where your secret weapon is the fourth dimension...

In the center of an orbiting space station you are protected only by a revolving force shield. Enemy forces are advancing from all directions and begin to place killer satellites in orbit around your station. And then, look out for the meteors!

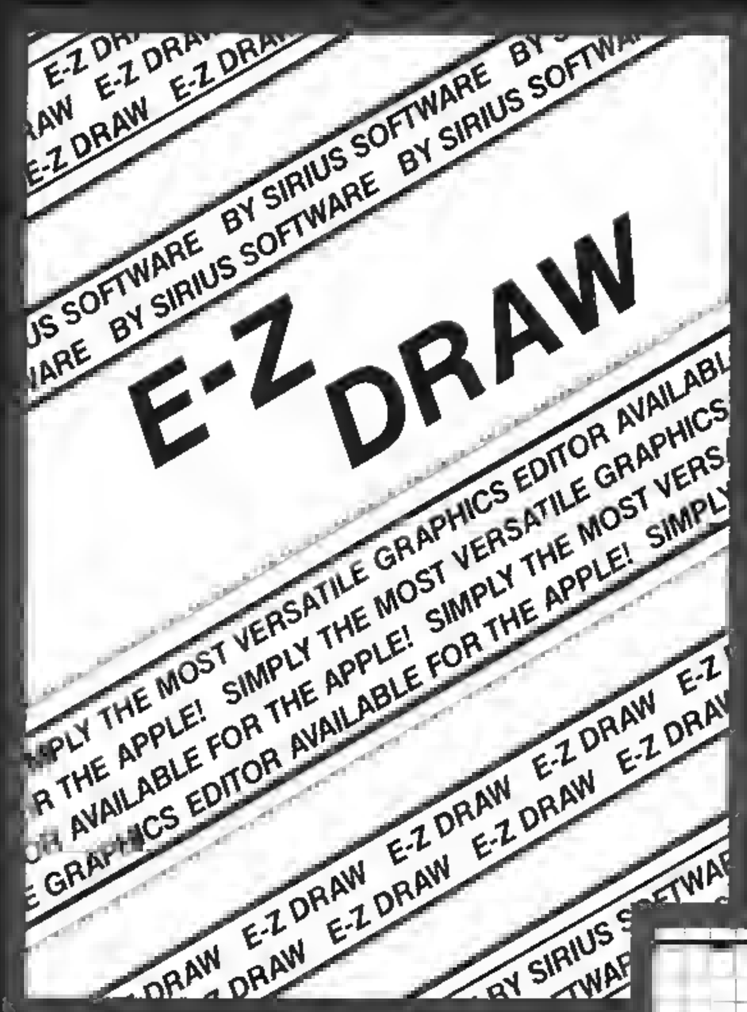
A "bloody" good game for the true-blue game freak. Your mission in this exploratory operation is to deliver whole blood to Hemophilia, a city in the sky, and return to Anemia Base before the Gamma Goblins overcome you. A real heart stopper!

What say we go out and stomp a few???

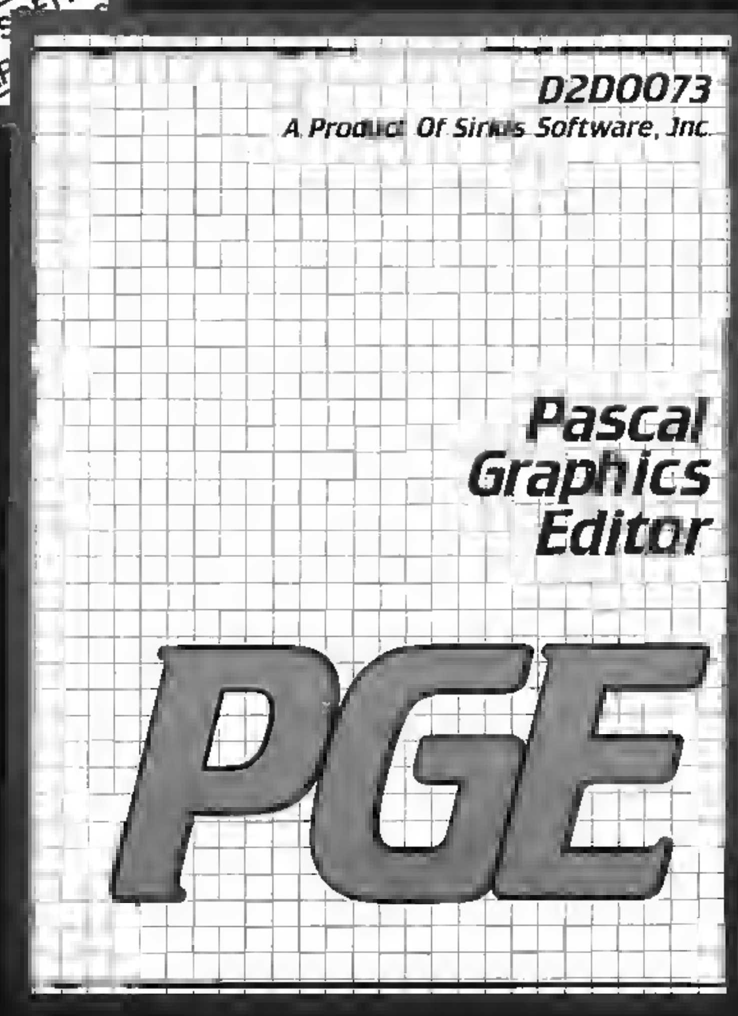
Endless Excitement Stomping Sneakers And A Swarm Of Other Creatures

Phantoms Five simulates a fighter-bomber mission in real time, three dimensional color graphics. While you try to make your bombing run, you have to avoid being hit by anti-aircraft fire and fight off enemy aircraft as well.

Hatch some fun with the Spiders, Wolves, Lips, and Fuzzballs. Space Eggs will crack you up! Each package includes a multi-color T-shirt iron-on that says "I FRIED THE SPACE EGGS."



This is the graphics editing package we based our business on. Includes the Higher Text Character Generator by Ron & Darrell Aldrich and over 20 original and imaginative type styles.



The professional graphics editing package for use within the Pascal environment.

BOTH BARRELS

A two game pack featuring "High Noon" and "Du Hunt" You'll love the bad guy that falls off the roof and the dogs fighting over the ducks. Fun for the young and the young at heart.
Copyright 1980 By Sirius Software, Inc.

STAR CRUISER

Save yourself from the swooping aliens! This is a f action arcade style game that can be played from ages three and up, but beware, the difficulty increases with each new wave of aliens.
Copyright 1980 By Sirius Software, Inc.

CYBER STRIKE

Interstellar challenge for the dedicated arcade ga You are in command of a light transport ship equi ped with Hyperspace Drive, Antimatter Torpedoes, Local and Galactic Sensors, Meteor Shields, and an Instrument Panel which continually tabulates all information vital to your mission. You alone ca prevent the clone take over of the allied settlemen bases. WARNING . this game requires practice to play successfully.
Copyright 1980 By Sirius Software, Inc.

AUTOBAHN

Hair raising excitement at 120, 160, and 200 kilom ters per hour! Drive through heavy traffic, oil slicks, narrow roads, and dark tunnels (with headlights). Watch out for the fire trucks! Only on the Autobahn can you drive this fast.
Copyright 1981 By Sirius Software, Inc.

PULSAR II

A unique two game series that provides scoring options for separate or combination game play. T destroy the "Pulsar" is no easy task. It is surrounded by spinning shields that send out orbs of energy aimed directly at you. "The Wormwall" places you one of the strangest mazes ever created. The walls not connect. Openings only occur temporarily as moving colored segments in the walls cross. In ad tion, there are munching mouthers in each level of the maze ready to gobble you up should you misjudge the time and location an opening will occ
Copyright 1981 By Sirius Software, Inc.

Contact Your Local Computer Dealer For More Information • Dealer Inquiries Invited

Sirius Software, In
2011 Arden Way #2, Sacramento, California 958

PROGRAMMING: Cops & Robbers was programmed by Alan Merrell and Eric Knopp. Epoch was programmed by Larry Miller. Orbitron was programmed by Eric Knopp. Gamma Goblins was programmed by Tony and Benny Ngo. E-Z Draw was programmed by Nasir Gebelli and Jerry W. Jewell. Pascal Graphics Editor was programmed by Ernie Brock. Sneakers was programmed by Mark Turmell. Gorgon, Phantoms Five, Space Eggs, Both Barrels, Star Cruiser, Cyber Strike, Autobahn, and Pulsar II were programmed by Nasir.

SYSTEM REQUIREMENTS: All software mentioned in this advertise require an Apple II or II+ with 48K with the following exceptions Draw requires a 48K Apple with Applesoft in ROM (or a 64K Apple II+) Pascal Graphics Editor requires an Apple II or II+ with Lan System.

MICRO

Microbes and Updates

Here's a note from A. Penaloza in Morton Grove, Illinois:

For those C2 and C4P users having the Indirect Jump Vectors Mod, where the absolute address of INPUT, OUTPUT, CTRL/c, SAVE and LOAD routines are transferred from ROM into RAM (see *PEEK 65*, July 1981), use the following variation of Mr. Piot's "Step and Trace" (MICRO 38:79):

Instructions — Same as Mr. Piot's except that five commands are recognized.

Control-S execute next instruction

Control-T display the line number

Control-U execute next instruction and display the line number

Control-C same as always

To turn on TRACE, execute the following:

POKE 11,68: POKE 12,02:
X = USR(X)

Note: When TRACE is on, Control-S or Control-U must be pressed for you to be able to execute anything.

Modified Single-Step and Trace

```
0000                  ;SINGLE STEP AND TRACE
0000                  ;
0000                         GETCHR = $FFB8
0000                         DISPLN = $B95A
0000                         CNTRLC = $FF99
0222                         * = $0222
0222 20B8FF  START    JSR GETCHR     ;CHARACTER IN ACC
0225 C902             CMP #$02       ;IS IT A CNTRL/B?
0227 F025             BEQ TROFF      ;YES, TURN OFF TRACE
0229 C903             CMP #$03       ;IS IT A CNTRL/C?
022B F014             BEQ RTN        ;YES, DO CNTRL/C
022D C913             CMP #$13       ;IS IT A CNTRL/S?
022F F010             BEQ RTN       ;YES, DO NEXT INSTRUCTION
0231 C914             CMP #$14       ;IS IT A CNTRL/T?
0233 D005             BNE CTRLU      ;NO, DO CNTRL/U
0235 205AB9           JSR DISPLN     ;YES, DISPLAY LINE #
0238 D0E8             BNE START      ;GET NEXT COMMAND
023A C915    CTRLU    CMP #$15       ;IS IT A CNTRL U?
023C D0E4             BNE START      ;NO, GET NEXT COMMAND
023E 205AB9           JSR DISPLN     ;YES, DISPLAY LINE #
0241 4C99FF  RTN      JMP CNTRLC    ;DO NEXT INSTRUCTION
0244 A922    TRON     LDA #$22       ;TURN ON TRACE, START LO
0246 8D1C02           STA $021C      ;VECTOR LO OF CNTROL/C
0249 A902             LDA #$02       ;ADDR HI OF START
024B 8D1D02           STA $021D      ;VECTOR HI OF CNTROL/C
024E 60               RTS
024F A999    TROFF    LDA #$99       ;TURN OFF TRACE, CTRL/C LO
0251 8D1C02           STA $021C
0254 A9FF             LDA #$FF       ;ADDR HI OF CNTROL/C
0256 8D1D02           STA $021D
0259 60               RTS
```

An update from Martin C. Foster of Virginia Beach, Virginia:

I have enclosed a listing of a modified version of the BASIC assembler program by Edward H. Carlson which you published in MICRO, March 1981.

I have modified it to run on the C1P and to accept hex numbers. Simply denote them by placing a "$" directly in front of the number. I have also modified the input routine so that commas may now be used. It still runs in 4K.

Modified Version of BASIC Assembler

```
0 FORX=1TO28:PRINT:NEXT
1 GOTO1990:REM ASSEMBLER
2 M1=INT(M/16):M2=M-M1*16:M1=FNH(M1):M2=FNH(M2)
3 Z=Z+1:POKEQ+Z,M1:Z=Z+1:POKEQ+Z,M2:RETURN
4 Z=Z+1:GOSUB2:POKEAD,M:AD=AD+1:RETURN
5 NI=INT(N/256):LO=N-256*HI:BY=3
7 II=INT(AD/256):JJ=AD-II*256:M=II:Z=1:GOSUB2
8 M=JJ:GOSUB2:M=OP:Z=Z+1:GOSU04
10 IFBY>1THENM=LO:GOSUB4
11 IFBY=3THENM=NI:GOSUB4
12 GOTO100
20 FORZ=1TOLEN(C$):POKEN+Z,ASC(MID$(C$,Z,1)):NEXT:RETURN
99 C$="ERROR":N=Q+5:GOSUB20
100 PRINTAD;:GOSUB1905:L$=LEFT$(C$,3):L=LEN(C$)
101 IFL>4THENC$=RIGHT$(C$,L-4):L=L-4:GOSUB110
102 IFL$="NEX"THENGOSUB4000:N=Q+5:GOSU020:GOTO100
103 IFL$="ADD"THENAD=VAL(C$):GOTO100
104 IFL$="CON"TNENCA=0:OP=VAL(C$):GOTO200
105 IFL$="DIS"THENAD=VAL(C$):OP=PEEK(AD):CA=0:GOTO200
106 IFL$="ASC"THENM=ASC(C$):Z=5:GOSUB2:GOTO100
109 GOTO124
110 IFASC(C$)=36THENC$=RIGNT$(C$,L-1):L=L-1:GOTO112
111 RETURN
112 C2$=RIGNT$(C$,2):IFC2$=",Y"ORC2$=",X"THEN114
113 GOSUB4000:L=LEN(C$):RETURN
114 C$=LEFT$(C$,L-2):GOSUB4000:C$=C$+C2$:L=LEN(C$):RETURN
```

(Continued)

```
115 C3$=RIGHT$(C$,3):C$=MID$(C$,2,L-4):L=LEN(C$)
116 IFC3$="),Y"ORC3$=",X)"THENGOSUB110:C$=C1$+C$+C3$
    :L=LEN(C$):RETURN
117 C$=C$+C3$:C3$=RIGHT$(C$,1):C$=LEFT$(C$,LEN(C$)-1):L=LEN(C$)
118 1FC3$=")"THENGOSU8110:C$=C1$+C$+C3$:L=LEN(C$)
I19 RETURN
124 FORI=1TO4:FORJ=1TO56STEP4
130 IFL$=MID$(C$(1),J,3)THENN=14*(I-1)+(J+3)/4:GOTO161
144 NEXTJ,I:GOTO99
161 CA=VAL(MID$(E$,N,1))
163 OP=VAL(MID$(F$(I),J,3))
200 BY=1:IFCA=0THEN7
210 IFC$="A"ANDCA=3THENOP=OP+8:GOTO7
213 IFC$="A"THEN99
220 BY=2:C1$=LEFT$(C$,1)
221 II=OP-8*(CA=1)
223 JJ=CA=1ORCA=4ORCA=5
224 1FC1$="#"ANDJJTHEN4060
228 IFC1$="#"TNEN99
230 IFC1$<>"("THEN260
231 GOSUB115: LO=VAL(MID$(C$,2,L-4))
232 IFRIGHT$(C$,3)="),Y"ANDCA=1TNENOP=OP+16:GOTO7
240 IFRIGHT$(C$,3)=",X)"ANDCA=1TNEN7
250 IFRIGHT$(C$,1)<>")"ORCA<>6TNEN99
254 N=VAL(MID$(C$,2,L-2))
258 OP=OP+32:GOTO5
260 IFRIGNT$(C$,2)<>",X"TNEN280
262 N=VAL(LEFT$(C$,L-2))
264 IFN>255THEN274
266 LO=N:IFCA=2TNENOP=OP+16:GOTO7
268 IFCA=1ORCA=3ORCA=5TNENOP=OP+20:GOTO7
269 GOTO99
274 IFCA=2THENOP=OP+24:GOTO5
276 1FCA=1ORCA=3ORCA=5TNENOP=OP+28:GOTO5
278 GOTO99
280 1FR1GNT$(C$,2)<>",Y"TNEN300
282 N=VAL(LEFT$(C$,L-2))
284 IFN>255THEN292
286 LO=N
287 IFCA=2ORCA=5THENOP=OP+I6-4*(CA=5):GOTO7
292 IFCA=1ORCA=5TNENOP=OP+24-4*(CA=5):GOTO5
299 GOTO99
300 N=VAL(C$)
305 IFCA=8THEN340
310 IFN>255THEN332
312 LO=N
3I4 1FCA=2ORCA=7THEN7
316 IFCA=1ORCA=3ORCA=4ORCA=5THENOP=OP+4:GOTO7
332 IFCA=2ORCA=7THENOP=OP+8:GOTO5
334 IFCA=1ORCA=3ORCA=4ORCA=5THENOP=OP+12:GOTO5
336 IFCA=6ORCA=9TNEN5
339 GOTO99
340 N=N-AD-2:IFN<-128ORN>127THENPR1NT"CAN'T 8RANCH";N:GOTO100
342 IFN<0THENN=N+256
344 LO=N:GOTO7
1900 RETURN
1905 C$=""
1910 X=USR(X):L=PEEK(531):PRINTCNR$(L);
1920 IFL=13TNENPRINTCNR$(10):RETURN
1930 C$=C$+CHR$(L):GOTO1910
1990 DIMC$(5),F$(5):POKE11,0:POKEI2,253
1992 DEFFNH(D)=D+48-7*(D>9)
1995 Q=54088
2000 C$(1)="ADC AND ASL BCC BCS 8EQ B1T 8MI BNE 8PL BRK 8VC BVS CLC"
2012 C$(2)="CLD CLI CLV CMP CPX CPY DEC DEX DEY EOR INC INX INY JMP"
2013 C$(3)="JSR LDA LDX LDY LSR NOP ORA PHA PNP PLA PLP ROL ROR RT1"
2016 C$(4)="RTS SBC SEC SED SEI STA STX STY TAX TAY TSX TXA TX3 TYA"
2020 E$="11388788808800001442001200691553010000330010001220000000"
2021 F$(1)="097 033 002 144 176 240 036 048 208 016 000 080 112 024"
2023 F$(2)="216 088 184 193 224 192 198 202 I36 065 230 232 200 076"
2025 F$(3)="032 161 162 160 066 234 001 072 008 104 040 034 098 064"
2027 F$(4)="096 225 056 248 120 129 134 132 I70 168 I86 138 154 152"
2029 G$="0123456789ABCDEF"
2030 AD=546:GOTO100
4000 N=0:L=I:FOR1=ITOLEN(C$)
4020 M=ASC(RIGHT$(C$,I))-48:1FM>9THENM=M-7
4040 N=N+M*L:L=L*16:NEXT
4050 C$=STR$(N):RETURN
4060 1FM1D$(C$,2,1)="$"THEN4080
4070 LO=VAL(RIGHT$(C$,L-1)):OP=11:GOTO7
4080 C$=RIGNT$(C$,L-2):GOSUB4000
4090 LO=VAL(C$):OP=II:GOTO7
```

N.P. Herzberg of Princeton, New Jersey, sent us this microbe:

I just received the August issue and was pleased to see my article, "Sorting with Applesoft," on pages 92 - 94. However, there were several errors in the listing:

Line 555: Change

    UP = I * I

to:

    UP = I : I

Line 750: Change

    R(J) = JS(TEMP) =

to:

    R(J) = J : S(TEMP) =

Line 2000: Change

    RENS OR T

to:

    REM SORT

Line 5050 is a three-line DATA statement. The first line is correct. The second line begins:

    3, 133, 132, ...

It should read:

    3, 133, 133, 132, ...

The third line ends:

    ... 16,143,96,0.

It should read:

    16, 243, 96, 0.

If you have a fix to a microbe or an updated version of an old program, share your knowledge. Send it to

Microbes & Updates
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

MICRO

New Publications

(Continued from page 36)

Computer Literacy

Computer Literacy: Problem-Solving with Computers by Carin E. Horn and James L. Poirot. Sterling Swift Publishing Company (1600 Fortview Road, Austin, Texas 78704), 1981, viii, 304 pages, photographs, diagrams, 7¼ × 9 1/8 inches, paperbound.
ISBN: 0-88408-133-8 $13.95

This is a textbook for classroom use at the high school or possibly junior high level.

CONTENTS: *Introduction; Computer Jargon; The History of Computing; Computer Applications; Computers in Government; The Value of Information in Society; Computer-Related Occupations; Computers and Humans; Computer Systems; Computer Components; Algorithms and Flowcharting; Computer Programming and Design Logic; Beginning BASIC; Bibliography; Glossary; Index.*

Owning Your Home Computer: The Complete Illustrated Guide by Robert L. Perry (Everest House Publishers (1133 Avenue of the Americas, New York, New York 10036), 1980, 224 pages, photographs, diagrams, 7 3/8 × 10 inches, paperbound.
ISBN: 0-89696-093-5 $10.95

This introductory work on personal computing for the layman not only covers common home computer applications, but focuses extensively on the home computer as an instrument for communicating with other computers and with data banks. In this area, the subjects covered include Teletext, Viewdata, QUBE, EIES, and DIGICAST.

CONTENTS: *The World at Your Fingertips*—The Home Information Explosion; What Is a Home Computer?; The First Generation; Chips off an Old Block; How to Buy a Home Computer. *The New Generations—1980 and Beyond*—The Newest Home Computers; The Handiest Home Computers; Putting the World at Your Fingertips—Easily; The Mind Appliance: The Once and Future Computer. *What Do You Do with a Mind Machine?*—Ninety-nine Common Things to Do with a Home Computer; The Three Rs and a C; Division of Labor: Home Computers in Your Work; The Next Step Beyond: An Introduction to Home Computer Programming; Help for the Handicapped; Mother's and Father's Little Helper. *The Thinking Computer of the Future*—The Thinking Computer of the Future. *Appendix*—1,050 Home Computer Programs. *Glossary; Bibliography; Index.*

A Welcome to PET Users

By Loren Wright

Every Issue

Because of our PET feature this month many of you PET owners will be reading MICRO for the first time. While there is normally only one PET article per issue, you will find there is more to MICRO than articles. For instance, there's the PET Vet column, which I have been writing regularly now for more than a year. It includes product announcements and other news useful to PET owners, programming and hardware notes, answers to reader questions and occasional product reviews. Other departments, although not specifically PET oriented, offer information for PET and other microcomputer users.

Unlike other magazines, MICRO is aimed at readers with more intermediate computer ability. MICRO readers are generally very comfortable with BASIC, and many are accomplished in assembly language. In the coming months we will be making more of an effort to convert novices to intermediates, by presenting more tutorial-type articles on higher level languages, structured programming, and concepts of assembly language programming.

This Issue

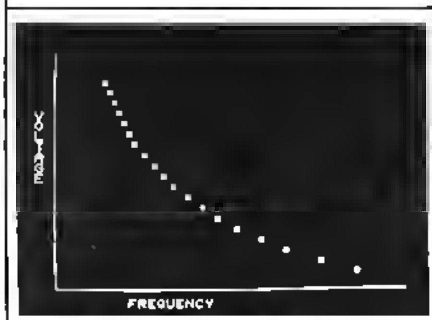
This issue offers a variety of articles for the CBM computer family. David Malmberg (author of "PRINT USING for the PET") discusses how to make a light pen work with the VIC and presents two demonstration programs. Programming a light pen is made particularly easy by the VIC's CRT controller and there are now two light pens available that work on the VIC.

Albert Reuss has compiled a series of tables that show all the various ROM configurations of Commodore products — no more question as to what kind of PET you have! In "The PET from A to D," John Sherburne describes the use of two different inexpensive devices that can convert analog signals to digital signals, which are understandable to the PET. Incidentally, there are two other articles in the issue dealing with analog to digital conversion for microcomputers.

The fourth article (which I wrote) covers how the PET handles character information, and presents three ideas for substitute character sets, which may be applied on PET, VIC, or OSI systems. Most PETs can have their character ROMs directly replaced by a custom EPROM, and with the decrease in EPROM prices in recent years, EPROM programming is now within the reach of nearly everyone.

ROM Genealogy · p. 50

VIC Light Pen · p. 54



PET from A to D · p. 60

Substitute Characters · p.64

Previous Issues

Following is a list of the PET articles that have appeared in MICRO back to December, 1980:

Future Issues

The future offers many interesting articles. A sampling: Tiny Pilot in machine language, memory protection for old PETs, a useful sound device that lets you listen to tapes load, and an audible disk alarm.

I believe MICRO has a lot to offer the PET, CBM, or VIC owner. If you want to learn more about other high level languages to improve your programming techniques, to get ideas for applications, to learn more of the workings of your computer, or to build useful add-on devices, then MICRO should be among the magazines you read regularly.

MICRO


JINSAM™ 8.0
space
age
micro
software
USA
Used at NASA,
Kennedy Space Center
With Multiple Applications Related
to the Columbia Space Shuttle Project
INQUIRY WELCOME.
JINI MICRO-SYSTEMS, Inc.
DATABASE MANAGEMENT SYSTEM DESIGN
BOX 274 KINGSBRIDGE STN., RIVERDALE, N.Y. 10463 (212) 796-6200

Commodore ROM Genealogy

What kind of PET do you have? Use these tables to find the ROM configuration, logic board, and character ganerator of your PET, CBM, disk drive, or printer. The author presents a brief history of Commodore configurations.

Albert I. Reuss
Box 151
Berkeley, California 94701

When the Commodore Personal Electronic Transactor (PET) first went into production in September 1977, it incorporated BASIC Level I. Some PETs used the 6540 28-pin ROM by MOS Technology, Inc., and others used the more standard 2316B 24-pin ROM.

The next up-grade production was to BASIC Level II. This corrected an intermittent bug in the edit software, and improved the garbage collection.

The next two production ROM sets were BASIC Level III, which allowed use of the Commodore disk drive. It also cleared up a bug which limited the dimensions to 256. At this time, the CBM "Professional Computers" with larger keyboards came into being. One set of ROMs was for the graphic (PET) keyboard CBM, and the other was for the business keyboard CBM.

Series 2001
Static RAM Versions

	6540 (28-pin) ROM		2316B (24-pin) ROM	
	ROM 1.0 BASIC Level I		**ROM 1.0 BASIC Level I**	
Location	**ROM**	**Part Number**	**ROM**	**Part Number**
H1	6540-011	901439-01	901447-01	901447-01
H2	6540-013	901439-02	901447-03	901447-03
H3	6540-015	901439-03	901447-05	901447-05
H4	6540-016	901439-04	901447-06	901447-06
H5	6540-012	901439-05	901447-02	901447-02
H6	6540-014	901439-06	901447-04	901447-04
H7	6540-018	901439-07	901447-07	901447-07
A2	6540-010	901439-08	901447-08	901447-08
Logic Board		320008		320081
	ROM 2.0 BASIC Level II		**ROM 2.0 BASIC Level II**	
H1	6540-019	901439-09	901447-09	901447-09
H2	6540-013	901439-02	901447-03	901447-03
H3	6540-015	901439-03	901447-05	901447-05
H4	6540-016	901439-04	901447-06	901447-06
H5	6540-012	901439-05	901447-02	901447-02
H6	6540-014	901439-06	901447-04	901447-04
H7	6540-018	901439-07	901447-07	901447-07
A2	6540-010	901439-08	901447-08	901447-08
Logic Board		320132		320137
	ROM 3.0 Up-Grade Retrofit BASIC Level III		**ROM 3.0 Up-Grade Retrofit BASIC Level III**	
H1	6540-020	901439-13	901465-01	901465-01
H2	6540-022	901439-15	901465-02	901465-02
H3	6540-024	901439-17	901447-24	901447-24
H4	6540-025	901439-18	901465-03	901465-03
H5	6540-021	901439-14	Blank	
H6	6540-023	901439-16	Blank	
H7	6540-026	901439-19	Blank	
A2	6540-010	901439-08	901447-08	901447-08
Logic Board		320132 or 320008		320137 or 320081

The next up-grade , known as BASIC Level IV, adds disk commands to the BASIC and further improves garbage and string handling. This has since been up-graded from ROM 4.0 to 4.1 to correct minor errors in the 4.0

At the same time, Commodore brought out the new 80-column business machines with some additional word-processing functions built in.

The 40-column business machines were dropped from production in 1981. 8K units were also dropped. However, there was a limited production of 3.0 BASIC, small keyboard, dynamic 8Ks.

The current production graphic keyboard model uses the legible 12", 40-column screen. It uses the same logic board that the 8000 series uses, and includes built-in sound, and a repeat key. All 40-column machines prior to this model used the 9" screen.

BASIC Level I through III are known as the 2001 Series. BASIC Level IV with 40-column screen is known as the 4000 series.

There have been three different character generator ROMs installed over these generations. In the early production runs through BASIC Level II, location A2 contained either a 6540 or 2316B ROM.

In BASIC Level III and IV in lcoation F10, you have 901447-10 (p/n 901447-10).

The 901447-10 ROM can replace the 901447-08 ROM in the up-grade from BASIC II to BASIC III. There is no replacement ROM for the 6540-010 28-pin ROM.

The 2023 Printer was discontinued in 1980. This was the friction feed model of the printer. The 2022 Printer (traction feed) was replaced by the 4022 Printer. The VIC-20 Color Computer was introduced in 1981.

This information has been compiled from a number of sources, including *Cursor #18*; and *Commodore Newsletter*, Vol. 1, No. 10.

Dynamic RAM Versions
Graphic Keyboard
ROM 3.0 — BASIC Level III
Series 2001

Hex Address	Location	ROM	Part Number
$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	Blank	
$C000	D6	901465-01	901465-01
$D000	D7	901465-02	901465-02
$E000	D8	901447-24	901447-24
$F000	D9	901465-03	901465-03
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.0 — BASIC Level IV
Up-grade retrofit 3.0 to 4.0 ROM's
Series 2001

$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-19	901465-19
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-29	901447-29
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.0 — BASIC Level IV
Series 4000

$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-19	901465-19
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-29	901447-29
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.1 — BASIC Level IV
Series 2001 & 4000

$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-23	901465-23
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-29	901447-29
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.1 — BASIC Level IV
Series 4000 — 12" Screen

$F000	UD6	901465-22	901465-22
$E000	UD7	901499-01	901499-01
$D000	UD8	901465-21	901465-21
$C000	UD9	901465-20	901465-20
$B000	UD10	901465-23	901565-23
$A000	UD11	Blank	
$9000	UD12	Blank	
Character Generator	UA3	901447-10	901447-10
Logic Board			8032030 or 8032080

Business Keyboard

ROM 3.0 — BASIC Level III
Series 2001

Hex Address	Location	ROM	Part Number
$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	Blank	
$C000	D6	901465-01	901465-01
$D000	D7	901465-02	901465-02
$E000	D8	901447-01	901447-01
$F000	D9	901465-03	901465-03
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.0 — BASIC Level IV
Up-grade retrofit 3.0 to 4.0 ROM's
Series 2001

Hex Address	Location	ROM	Part Number
$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-19	901465-19
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-02	901447-02
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.0 — BASIC Level IV
Series 4000

Hex Address	Location	ROM	Part Number
$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-19	901465-19
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-02	901447-02
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

ROM 4.1 — BASIC Level IV
Series 2001 & 4000

Hex Address	Location	ROM	Part Number
$9000	D3	Blank	
$A000	D4	Blank	
$B000	D5	901465-23	901465-23
$C000	D6	901465-20	901465-20
$D000	D7	901465-21	901465-21
$E000	D8	901447-02	901447-02
$F000	D9	901465-22	901465-22
Character Generator	F10	901447-10	901447-10
Logic Board			320351

VIC-20

Location	ROM	Part Number
D5	901486-01	901486-01
D6	901486-06	901486-06
C7	901460-03	901460-03
		1001008

PRINTERS

Location	ROM	Part Number
2022 Printer (continuous feed)		
U11	901472-03	901472-03
	Logic Board	320311
2023 Printer (friction feed)		
U11	901472-02	901472-02
	Logic Board	320311
2023 Printer Interim Fix		
U11	901472-03	901472-03
	Logic Board	320311
2022 & 2023 Printers Interim Fix		
U11	901472-04	901472-04
	Logic Board	320311
2022 Printer ??? Fix ???		
U11	901472-07	901472-07
	Logic Board	320311
4022 Printer		
U11	901472-07	901472-07
	Logic Board	4022004

Series 8000 — 12" Screen
ROM 4.0 — BASIC Level IV

Hex Address	Location	ROM	Part Number
$F000	UD6	901465-22	901465-22
$E000	UD7	901474-03	901474-03
$D000	UD8	901465-21	901465-21
$C000	UD9	901465-20	901465-20
$B000	UD10	901465-19	901565-19
$A000	UD11	Blank	
$9000	UD12	Blank	
Character Generator	UA3	901447-10	901447-10
Logic Board			8032030 or 8032080

Series 8000 — 12" Screen
ROM 4.1 — BASIC Level IV

Hex Address	Location	ROM	Part Number
$F000	UD6	901465-22	901465-22
$E000	UD7	901474-03	901474-03
$D000	UD8	901465-21	901465-21
$C000	UD9	901465-20	901465-20
$B000	UD10	901465-23	901465-23
$A000	UD11	Blank	
$9000	UD12	Blank	
Character Generator	UA3	901447-10	901447-10
Logic Board			8032030 or 8032080

DISKS

DOS 1.0 — 2040 Disk

Location	ROM	Part Number
UL1	901468-06	901468-06
UK1	Blank	
UH1	901468-07	901468-07
UK3	901466-02	901466-02
UK6	901467	901467
	Logic Board	320820
	Logic Board	320817

DOS 2.1 — 4040 Disk
Up-grade retrofit 2040 to 4040

Location	ROM	Part Number
UL1	901468-12	901468-12
UK1	901468-11	901468-11
UH1	901468-13	901468-13
UK3	901466-04	901466-04
UK6	901467	901467
	Logic Board	320820
	Logic Board	320817

DOS 2.5 — 8050 Disk

Location	ROM	Part Number
UL1	901482-03	901482-03
UH1	901482-04	901482-04
UK3	901483-03	901483-03
UK6	901467	901467
	Logic Board	8050002
	Logic Board	8050006

MICRO

VIC Light Pen-manship

The video interfece chip used in the VIC constently keeps treck of the position of e light pen, meking softwere for it easy to write. There ere two light pens eveileble thet work with the VIC. This euthor expleins how they work, end gives two demonstration progrems: "Light Pen Scribe" and "Light Pen Artist".

David Malmberg
43064 Via Moraga
Fremont, California 94538

Most implementations of light pens on personal computers use an artificial technique to simulate a real light pen. This is done by setting up a table of possible pen screen locations that are to be tested and blinking them on and off very quickly with a lighted cursor space. The light pen is able to detect the change in light caused by the blink. By matching the pen's positive reading against which specific location is being blinked at that moment, the computer is able to determine the correct screen location. This technique has a number of significant drawbacks. For instance, it requires a great deal of memory, programming effort, and processor time to set up and test even a limited number of locations. If you want to test a large number of possible locations, the benefits seldom justify the necessary effort or the design compromises required.

The new Commodore Video Interface Chip (VIC) has alleviated this problem. Now, personal computer owners can have a *real* light pen whose location on the screen can be determined by the hardware — not by software gimmicks. The VIC light pen can detect any point on the screen instantly and automatically. It does not require time- and memory-consuming table look-ups and individual screen location testing. A light pen on the VIC is fun and easy to use; it can produce dramatic enhancements to games, educational programs, and menu-driven applications.

The VIC owner has several light pen options. The VIC was specifically designed to work with the Atari light pen (as well as the Atari joystick and game paddles). The Atari light pen retails for about $75. Systems Formulate is also marketing a light pen which was developed in Japan and costs about $35. The authoritative voice on the other end of the Commodore telephone "Hotline" said that Commodore will have its own VIC light pen on the market by Christmas (price unknown). Their pen will be essentially identical to the Atari pen. All of these pens work the same way and software written for any one pen should run using the others (with only one slight difference which will be described later).

How It Works

The VIC was specifically designed to handle color video graphics on a monitor or home TV. This chip shares the workload with a 6502 chip which handles most of the processing and the operating system. This design philosophy is similar to that of the Atari computer which also has a 6502 for its main processor and a separate chip to handle its video.

Among the functions this special video chip performs is controlling and tracking the electronic beam that actually "paints" the picture on the screen. This beam sweeps from left to right across the screen and from top to bottom — painting a completely new picture on the screen 60 times each second. This speed is certainly faster than the eye can detect, but slow enough so that the VIC can track the beam's location as it moves through every dot on the screen.

To demonstrate just how the VIC knows the exact location of the beam, let's assume you have plugged your light pen into the VIC game port and have entered and run the following short program:

```
10 X = PEEK(36870)
20 Y = PEEK(36871)
30 LT = -((PEEK(37151)AND4)
     =0)
40 PRINT "CLEAR"X;Y;LT
50 GOTO 10
```

As the electronic beam moves around the screen and the light pen senses the light from the beam, the VIC captures the screen's horizontal and vertical coordinates at that instant and stores them in locations 36870 and 36871, respectively.

As you experiment by running your light pen over the screen surface you should notice the following points:

1. The value of X will vary from approximately 30 on the extreme left side of the screen to approximately 122 on the extreme right. Similarly, Y will vary from about 17 at the top of the screen to about 121 at the bottom. Your own readings may differ slightly from these. The X and Y values change linearly with the corresponding horizontal and vertical movement of the pen.

2. Inside the screen border, X varies from 32 on the left to 119 on the right — a length of 88 (counting zero as the left-most value). This corresponds with 4 light pen values per screen space (there are 22 columns). The Y values vary from 24 at the top border to 115 at the bottom border. This is a length of 92 light pen values or 4 for each of the 23 screen rows.

3. If you are using an Atari light pen or (presumably) its Commodore equivalent, you should notice that the value of LT changes from 0 to 1 whenever the tip of the pen is pressed against the screen. The Atari pen has

a spring-loaded switch in its tip which can be read by the VIC using statement 30 above. This is a very nice feature and will enhance your light pen applications.

4. You will also notice that the pen is probably more sensitive than you might have imagined. For example, the Atari light pen I use can be detected as far as six inches away from the screen. And yet, it can be controlled fairly accurately even from that distance. It makes you wonder when someone might come out with a light "gun" that could be "shot" at the screen from several feet away. Just think of the games you could write!

5. Even though the pen is fairly sensitive, it is subject to "noise." As a demonstration, if you try to hold the pen absolutely still in one place, you will see the values of X and Y flickering to nearby values — and occasionally to a more distant value. This is caused by "noise" in the pen. This "noise" seems to be partly a function of color. Reading a location that is a dark color, especially if nearby locations are light colors, may cause problems. One of the routines I will present later will solve these problems.

6. The last thing you should notice is that the pen always shows the last value of X and Y that it read. If you take the pen completely away from the screen it still indicates a specific location. Because of this, you should be very careful in the design of your applications; do not mistake an old pen reading for a new one. This potential problem is another good reason for the Atari's tip switch.

Light Pen Scribe

As an example of how the light pen might be used in a VIC program, let's examine the BASIC program in listing 1. This program displays the normal upper case character set including numbers and punctuation symbols on the top three lines of the VIC screen. The next two lines contain normal "cursor" control words such as CLEAR, HOME, RETURN, UP, etc. The sixth line displays the seven color options (excluding white) available for the VIC. By putting the light pen on various characters, cursor commands, or colors in these first six lines, the program generates written text in whatever color combinations you wish on the remainder of the VIC screen. In essence, the program totally replaces keyboard input with light pen input.

Listing 1

```
100 REM LIGHT PEN SCRIBE
110 REM BY DAVID MALMBERG
120 DIM CC(7) : POKE36879,27 : CL=3 : SS=7680 : IC=30720
130 DEF FNA(Z)= SS+CN+22*RW : DEF FNB(Z)= FNA(Z)+IC :
    DEF FNC(Z)=PEEK(FNA(Z))
140 FORI=1TO7 : READ CC(I) : NEXT
150 DATA 144,28,159,156,30,31,158
160 PRINT"J"
170 FOR I=0TO63 : POKESS+I,I : POKESS+IC+I,CL : NEXT
180 PRINT"HOME RETURN UP DONN"
190 PRINT"CLEAR ERASE LEFT RIGHT";
200 FOR I=1TO7 : PRINTCHR$(CC(I))"   ";  : NEXT
210 RW=6 : CN=0
220 IF CN<0 THEN CN=21 : RW=RW-1
230 IF CN>21 THEN CN=0 : RW=RW+1
240 IF RW<6 THEN RN=6
250 IF RW>22 THEN RW=22
260 IF FNC(0)=32 THEN POKE FNB(0),CL
270 IF FNC(0)<128 THEN POKE FNA(0),FNC(0)+128
280 REM TEST LIGHT PEN TIP SWITCH
290 LT=-((PEEK(37151)AND4)=0) : IF LT=0 THEN 290
300 REM SWITCH ON - NOW READ AND TRANSLATE PEN LOCATION
310 X=PEEK(36870) : Y=PEEK(36871)
320 IF X<32 THEN X=32
330 IF X>119 THEN X=119
340 IF Y<24 THEN Y=24
350 IF Y>115 THEN Y=115
360 X=X-32 : Y=Y-24
370 C=INT(X/4) : R=INT(Y/4)
380 L=SS+C+22*R : LC=PEEK(L)
390 REM CHECK COMMAND - TAKE ACTION
400 IF R>5 THEN 290
410 IF R<5 THEN 470
420 REM COLOR ROW
430 X=INT((C+1.1)/3) : Y=(C+1)/3
440 IF X<>Y THEN 290 :REM INVALID COLOR LOCATION
450 IF X=1 THEN X=0
460 POKE 36879,24+X : CL=X : GOTO640
470 IF R<4 THEN 540
480 REM ROW 4 COMMANDS
490 IF C=0 THEN POKE FNA(0),FNC(0)-128 : GOTO 160 : REM CLEAR
500 IF C=6 THEN POKE FNA(0),32 : CN=CN-1 : GOTO 640 : REM ERASE
510 IF C=12 THEN POKE FNA(0),FNC(0)-128 : CN=CN-1 : GOTO640 :
    REM RIGHT
520 IF C=17 THEN POKE FNA(0),FNC(0)-128 : CN=CN+1 : GOTO640 :
    REM LEFT
530 GOTO 290 : REM INVALID LOCATION
540 IF R<3 THEN 620
550 REM ROW 3 COMMANDS
560 IF C=0 THEN POKE FNA(0),FNC(0)-128 : CN=0 : RN=6 : GOTO640 :
    REM HOME
570 IF C=5 THEN POKE FNA(0),FNC(0)-128 : CN=0 : RN=RW+1 :
    GOTO640 : REM RETURN
580 IF C=12 THEN POKE FNA(0),FNC(0)-128 : RN=RN-1 : GOTO640 :
    REM UP
590 IF C=15 THEN POKE FNA(0),FNC(0)-128 : RW=RW+1 : GOTO640 :
    REM DOWN
600 GOTO290 :REM INVALID LOCATION
610 REM WRITE CHARACTER AT PEN LOCATION
620 POKEFNA(0),LC : POKEFNB(0),CL : CN=CN+1
630 REM FLASH PEN LOCATION
640 S=0 : IF LC>127 THEN S=LC : LC=LC-128
650 POKE36878,15 : FOR I=1TO10
660 POKEL,LC+128 : POKE36876,225
670 POKEL,LC : POKE36876,195
680 NEXT : IF S<>0 THEN POKEL,S
690 POKE36878,0 : POKE36876,0 : GOTO220 READY.
```

By way of further explanation, here are some of the functions being performed in various parts of the program.

Line 120 sets the color combination to a white screen with a light blue border by POKE36879,27. CL is the variable denoting the current color of the text and of the screen border. CL is initialized to light blue by setting it to 3. SS is the address of the start of screen memory (equivalent to 32768 on the PET). IC is the constant you must add to a particular screen location to get that location's corresponding color matrix location. For example, if we POKEd 7680 with a 1 (an A), we would also have to POKE 7680 + IC with a 3 to make that "A" light blue.

Line 130 sets up three functions: FNA returns the screen location corresponding to row RW and column CN; FNB gives the corresponding color matrix location; FNC returns the PEEKed value of FNA.

Lines 140 and 150 read the character values corresponding to various color print commands. For example, the third value of the array CC contains the character value that, when printed as CHR$(CC(3)), would cause the material printed next to be printed in light blue.

Lines 170 to 200 print the characters, commands and color options in the top six rows of the screen.

Lines 220 to 250 assure that the location where the text is being printed on the bottom part of the screen (current row RW and current column CN) is always within the proper bounds. These lines also assure that the text properly wraps around from the end of one row to the beginning of the next.

Lines 260 and 270 print the "cursor" where the text is to be written next.

Line 290 waits until the tip switch on the light pen has been depressed. If you are using a light pen that does not have a tip switch (or something similar), you should substitute the following line:

```
290 A$ = " " : GET A$ : IF
    A$ = " " THEN 290
```

This line will let you indicate that you are ready to have the light pen read a location by just touching any key on the keyboard. This is obviously a less elegant approach than using a tip switch.

Lines 310 to 380 read the current light pen horizontal and vertical values and translate them into the appropriate screen row and column. The variables L and LC are the screen location that the pen is pointing to, and the character at that location, respectively.

Lines 430 to 460 edit color commands from the pen and change the text and border colors accordingly.

Lines 490 to 600 edit "cursor" control commands and cause the proper action to be taken.

Line 620 writes the character (that the light pen is pointing to in the top three lines of the screen) where the current "cursor" position is on the bottom text area of the screen.

Lines 640 to 690 flash the location the pen is pointing to on and off, and sound a "buzzer" to indicate that the location was actually read, passed the edit, and was processed properly. Then the program loops back to line 220 and waits for the next light pen reading.

Light Pen Artist

Listing 2 contains another light pen application called "Light Pen Artist." It is a program that enables you to use the light pen to "paint" on the screen using

the VIC graphic character set and a pallette of seven colors. You can paint a single screen location or paint a "line" of the current graphic character between any two points on the screen.

The program works like the previously described "Light Pen Scribe." The top three lines on the screen contain the graphic characters. The fourth line bas these command words: CLR, POINT, LINE and (reverse) ON or OFF. The fifth line has the seven colors in the VIC artist's pallette. By simply pointing the pen at the color, the command options, and the character, you are ready to paint on the bottom portion of tbe screen by using your pen as if it were an artist's brush. It is easy, fun and a very impressive demonstration of the VIC's capabilities.

The basic program structure and even the variable names are almost identical to "Light Pen Scribe."

Noise Elimination

The artist program does have one significant difference. Lines 660 to 970 contain a machine language program (in the form of BASIC POKEs) that eliminates the "noise" from the pen readings. This is a general subroutine that could be appended to and used in other light pen programs. It loads into the VIC's cassette buffer, so you should be careful not to do any input/output that would destroy the routine when using it in other programs.

The best way to describe what tbe routine does is to look at the BASIC code in lines 310 to 370 in the Scribe program (listing 1). The machine language code performs the same calculations that these BASIC statements do to "normalize" tbe horizontal (X) and vertical (Y) values to begin with zero and not exceed 87 for X and 91 for Y. It also calculates the pen's column and row (also beginning with zero). All of these values are obtained by PEEKing locations 982 to 985, respectively.

Besides just making the arithmetic faster, the routine eliminates the noise by taking seven separate readings (at 1 jiffy intervals so they will correspond to different sweeps of the electronic beam), sorting them, and returning tbe median reading (i.e., fourth). Taking the median reading eliminates the "noisy" readings, because when sorted, these strange readings would certainly be at one extreme (or the other) of the list. The median value, on the other hand, is almost guaranteed to be "noise" free.

Listing 2

```
100 REM LIGHT PEN ARTIST
110 REM BY DAVID MALNBERG
120 DIM CC(7) : POKE36879,27 : CL=3 :SS=7680 : IC=30720
130 DEF FNA(Z)=SS+CN+22*RN : DEF FNB(Z)=FNA(Z)+IC :
    DEF FNC(Z)=PEEK(FNA(Z))
140 FOR I=1TO7 : READ CC(I) : NEXT
150 DATA 144,28,159,156,30,31,158
160 GOSUB760
170 PRINT"■" : PC=32 : RV=0 : LN=0 : LX=0 : LY=0
180 FOR I=64TO127 : POKESS+I-64,I : POKESS+I+IC-64,CL : NEXT
190 PRINT"■■■■C■LR ■P■OINT ■L■INE ■O■N ■O■FF"
200 FOR I=1TO7 : PRINTCHR$(CC(I))"  ■ ■"; : NEXT
210 REM TEST LIGHT PEN TIP SWITCH
220 LT=-((PEEK(37151)AND4)=0) : IF LT=0 THEN 220
230 REM SWITCH ON - NOW READ AND TRANSLATE PEN LOCATION
240 SYS(828) : CN=PEEK(984) : RN=PEEK(985)
250 L=FNA(0) : LC=FNC(0)
260 REM CHECK PEN LOCATION - TAKE APPRORRIATE ACTION
270 IF RW>4 THEN 520 : REM DRAN ROUTINE
280 IF RN<4 THEN 340
290 REM CHANGE COLOR
300 X=INT((CN+1.1)/3) : Y=(CN+1)/3
310 IF X<>Y THEN 220 : REM INVALID COLOR LOCATION
320 IF X=1 THEN X=0
330 POKE36879,24+X : CL=X : GOTO450
340 IF RW<3 THEN 430
350 REM ROW 3 COMMANDS
360 IF CN=0 THEN 170 : REM CLEAR
370 IF CN=4 THEN LN=0 : GOTO450 : REM POINT
380 IF CN=10 THEN LN=1 : GOTO450 : REM LINE
390 IF CN=15 THEN RV=1 : GOTO450 : REM REVERSE ON
400 IF CN=18 THEN RV=0 : GOTO450 : REM REVERSE OFF
410 GOTO220 : REM INVALID COMMAND LOCATION
420 REM UPDATE CURRENT CHARACTER
430 PC=LC
440 REM FLASH PEN LOCATION
450 S=0 : IF LC>127 THEN S=LC : LC=LC-128
460 POKE36878,15 : FOR I=1TO10
470 POKEL,LC+128 : POKE36876,225
480 POKEL,LC : POKE36876,195
490 NEXT : IF S<>0 THEN POKEL,S
500 POKE36878,0 : POKE36876,0 : GOTO220
510 REM DRAW ROUTINES USING CURRENT OPTIONS
520 V=PC : IF RV=1 THEN V=PC+128 : REM REVERSE IF APPROPRIATE
530 IF LN<>0 THEN 590
540 REM DRAW POINT
550 POKE FNA(0),V : POKE FNB(0),CL
560 LX=CN : LY=RN : REM UPDATE LAST POINT
570 GOTO220
580 REM DRAW LINE FROM LAST POINT TO CURRENT POINT
590 DX=CN-LX : DY=RW-LY
600 Z=ABS(DX) : IF ABS(DY)>Z THEN Z=ABS(DY)
610 IF Z=0 THEN IX=0 : IY=0 : GOTO630
620 IX=DX/Z : IY=DY/Z
630 FOR I=0TOZ : RN=INT(LY+IY*I+0.5) : CN=INT(LX+IX*I+0.5)
640 POKE FNA(0),V : POKE FNB(0),CL
650 NEXT : LX=CN : LY=RW : GOTO220
660 REM MACHINE LANGUAGE ROUTINE THAT TAKES 7 CONSECUTIVE
    READINGS
670 REM OF THE LIGHT PEN LOCATION AND SORTS THEN TO ELININATE
680 REM LIGHT PEN "NOISE" BY RETURNING THE MEDIAN READING
    (4TH OF 7).
690 REM THE READING IS "NORMALIZED" TO BEGIN NITH ZERO.
700 REM THE VALUES FOR THE READING CAN BE FOUND BY PEEKING
     THESE LOCATIONS:
710 REM     982 - HORIZONTAL VALUE (RANGES FROM 0 TO 87)
720 REM     983 - VERTICAL VALUE (RANGES FROM 0 TO 91)
730 REM     984 - SCREEN COLUMN (0 TO 21)
740 REM     985 - SCREEN ROW (0 TO 22)
750 REM****CAUTION****THE ROUTINE NILL BE DESTROYED IF
     ANY TAPE I/O IS DONE
760 FOR I= 828 TO 986 :READDC:POKEI,DC:NEXTI
770 DATA162,0,160,3,132,152,173,6,144
780 DATA201,32,176,2,169,32,201,120,144
790 DATA2,169,119,56,233,32,160,219,132
800 DATA151,32,179,3,165,151,24,109,218
810 DATA3,133,151,144,2,230,152,173,7
820 DATA144,201,24,176,2,169,24,201,116
```

(Continued)

Listing 2 *(Continued)*

```
 830 DATA144,2,169,115,56,233,24,32,179
 840 DATA3,232,236,218,3,240,9,165,162
 850 DATA197,162,240,252,76,62,3,173,218
 860 DATA3,74,168,177,151,141,215,3,169
 870 DATA219,133,151,169,3,133,152,177
 880 DATA151,141,214,3,173,214,3,74,74
 890 DATA141,216,3,173,215,3,74,74,141
 900 DATA217,3,96,142,214,3,172,214,3
 910 DATA192,0,240,22,136,209,151,200
 920 DATA176,16,136,141,214,3,177,151
 930 DATA200,145,151,136,173,214,3,56
 940 DATA176,230,145,151,96,0,0,0,0,7
 950 RETURN
READY.
```

The assembly source for this routine is given in listing 3. This routine is my adaptation for the VIC of a similar routine in the Atari Light Pen Operators Manual. The credit for the idea and the majority of the code should go to some anonymous programmer at Atari. The assembly source is very well commented and should be easy to follow. Pay particular attention to the logic of the "insert sort" in locations \$03B3 to \$03D5 of the source listing. This is a very clever routine that performs the sort as the data is being read, by making sure that each reading is inserted in its appropriate place in the table.

David Malmberg is Director of Management Systems for Foremost-McKesson in San Francisco. He has a PET, as well as a VIC, and is interested in machine language utilities, strategy games, and writing his own "Adventures." He'd like to hear from anyone who develops interesting VIC applications (with or without the light pen).

Listing 3

```
                    ;*******************************************
                    ;*
                    ;* VIC LIGHT PEN ROUTINE
                    ;*
                    ;* ADAPTED FOR THE VIC BY DAVID MALMBERG
                    ;* FROM A SIMILAR ROUTINE FOR THE ATARI
                    ;* AS DOCUMENTED IN THE ATARI LIGHT PEN MANUAL
                    ;*
                    ;*******************************************
                    TBLPTR      .DE $97
                    JIFFY.CTR   .DE $A2
                    PEN.HOR     .DE 36870
                    PEN.VER     .DE 36871
                    ;
                                .BA $33C         ; IN CASSETTE BUFFER
                                .OS
                    ;
033C- A2 00                     LDX #0           ; INITIALIZE COUNTER
033E- A0 03         LOOP        LDY #H, TABLES ; STORE TABLE HI-BYTE POINTER
0340- 84 98                     STY *TBLPTR+1
                    ;
                    ; LOAD X VALUE AND CONVERT
                    ;
0342- AD 06 90      X.LOAD      LDA PEN.HOR     ; GET LIGHT PEN X COORD
0345- C9 20                     CMP #32         ; PEN READING >= 32 ?
0347- B0 02                     BCS HERE        ; IF YES
0349- A9 20                     LDA #32         ; IF NO - SET TO 32 MINIMUM .
034B- C9 78         HERE        CMP #120        ; PEN READING =< 119 ?
034D- 90 02                     BCC HERE1       ; IF YES
034F- A9 77                     LDA #119        ; IF NO - SET TO 119 MAXIMUM
0351- 38            HERE1       SEC
0352- E9 20                     SBC #32         ; X VALUE NOW RANGES FROM 0 TO 87
                    ; CALL SORT ROUTINE
0354- A0 DB                     LDY #L,.TABLES
0356- 84 97                     STY *TBLPTR
0358- 20 B3 03                  JSR SORT
                    ;
                    ; FIND AND STORE ADDRESS OF YTABLE
                    ;
035B- A5 97                     LDA *TBLPTR
035D- 18                        CLC
035E- 6D DA 03                  ADC FRAMES      ; ADD TABLE SIZE TO X TABLE TO
0361- 85 97                     STA *TBLPTR     ; FIND YTABLE ADDRESS
0363- 90 02                     BCC Y.LOAD
0365- E6 98                     INC *TBLPTR+1
                    ;
                    ; LOAD Y VALUE AND CONVERT
                    ;
0367- AD 07 90      Y.LOAD      LDA PEN.VER     ; GET LIGHT PEN Y COORD
036A- C9 18                     CMP #24         ; PEN READING >= 24 ?
036C- B0 02                     BCS HERE2       ; IF YES
036E- A9 18                     LDA #24         ; IF NO - SET TO 24 MINIMUM
0370- C9 74         HERE2       CMP #116        ; PEN READING =< 116 ?
0372- 90 02                     BCC HERE3       ; IF YES
0374- A9 73                     LDA #115        ; IF NO - SET TO 115 MAXIMUM
```

(Continued)

```
0376- 38        HERE3     SEC
0377- E9 18               SBC #24         ; Y VALUE NOW RANGES FROM 0 TO 91

                ; CALL SORT ROUTINE
0379- 20 B3 03            JSR SORT
                ;
                ;REPEAT NUMBER OF TIMES SPECIFIED BY FRAMES
                ;
037C- E8                  INX
037D- EC DA 03            CPX FRAMES
0380- F0 09               BEQ FINISH
0382- A5 A2               LDA *JIFFY.CTR ; LOAD CURRENT JIFFY CYCLE
0384- C5 A2     WAIT      CMP *JIFFY.CTR ; TEST FOR NEXT JIFFY
0386- F0 FC               BEQ WAIT
0388- 4C 3E 03            JMP LOOP
                ;
                ;X,Y READING DONE - MOVE MEDIANS (CENTER VALUES)
                ; TO LOCATIONS "XCOORD" AND "YCOORD"
                ;
038B- AD DA 03  FINISH    LDA FRAMES     ; LOAD SIZE OF TABLE
038E- 4A                  LSR A          ; DIVIDE BY 2 FOR MEDIAN
038F- A8                  TAY            ; LOAD MEDIAN OFFSET IN Y REG
0390- B1 97               LDA (TBLPTR),Y ; LOAD Y MEDIAN
0392- 8D D7 03            STA YCOORD
0395- A9 DB               LDA #L, TABLES
0397- 85 97               STA *TBLPTR
0399- A9 03               LDA #H, TABLES
039B- 85 98               STA *TBLPTR+1
039D- B1 97               LDA (TBLPTR),Y ; LOAD X MEDIAN
039F- 8D D6 03            STA XCOORD
                ;
                ; CONVERT X,Y COORDS TO COLUMN AND ROW
                ;
03A2- AD D6 03            LDA XCOORD     ; LOAD X MEDIAN
03A5- 4A                  LSR A          ; DIVIDE BY 2
03A6- 4A                  LSR A          ; DIVIDE BY 2 AGAIN (IE, BY 4)
03A7- 8D D8 03            STA COL
03AA- AD D7 03            LDA YCOORD     ; LOAD Y MEDIAN
03AD- 4A                  LSR A          ; DIVIDE BY 2
03AE- 4A                  LSR A          ; DIVIDE BY 2 AGAIN (IE, BY 4)
03AF- 8D D9 03            STA ROW        ; STORE IN ROW
03B2- 60                  RTS            ; BACK TO BASIC
                ;
                ; PERFORM INSERT SORT OF READINGS INTO
                ; TABLE SPECIFIED IN TBLPTR
                ;
03B3- 8E D6 03  SORT      STX XCOORD     ; TRANSFER X REG TO Y REG
03B6- AC D6 03            LDY XCOORD     ; USING XCOORD AS TEMP STORAGE
03B9- C0 00     INSLOOP   CPY #0         ; HIT BOTTOM??
03BB- F0 16               BEQ INSERT     ; IF SO, INSERT
03BD- 88                  DEY            ; MOVE POINTER DOWN
03BE- D1 97               CMP (TBLPTR),Y ; COMPARE TO NEXT ENTRY
03C0- C8                  INY            ; RESTORE POINTER
03C1- B0 10               BCS INSERT     ; IF A>=M, NEW ENTRY'S SLOT FOUND
03C3- 88                  DEY            ; MOVE INDEX TO NEXT ENTRY
                ; MOVE Y TABLE ENTRY TO Y+1
03C4- 8D D6 03            STA XCOORD     ; TEMPORARILY SAVE ACC
03C7- B1 97               LDA (TBLPTR),Y
03C9- C8                  INY            ; MOVE POINTER FORWARD
03CA- 91 97               STA (TBLPTR),Y
03CC- 88                  DEY            ; RESTORE POINTER
03CD- AD D6 03            LDA XCOORD     ; RESTORE ACC
03D0- 38                  SEC
03D1- B0 E6               BCS INSLOOP    ; LOOP AGAIN
                ; NEW VALUE'S PROPER PLACE FOUND - DROP IT IN
03D3- 91 97     INSERT    STA (TBLPTR),Y
03D5- 60                  RTS            ; FROM SORT
                ;**********************************************
                ;*
                ;* RESULTS STORED HERE
                ;*
03D6- 00        XCOORD    .BY 0          ; X VALUE (0 TO 87)
03D7- 00        YCOORD    .BY 0          ; Y VALUE (0 TO 91)
03D8- 00        COL       .BY 0          ; SCREEN COLUMN (0 TO 21)
03D9- 00        ROW       .BY 0          ; SCREEN ROW (0 TO 22)
                ;*
                ;**********************************************
03DA- 07        FRAMES    .BY 7          ; NUMBER OF READINGS TAKEN
                TABLES    .EN
```

MICRO

The PET from A to D

The author describes two inaxpensive davices that can be used to convert analog signals to digital form. PET demonstration programs are included.

John Sherburne
4418 Andes Dr.
Fairfax, Virginia 22030

The idea of connecting real-world sensors to my PET and capturing outside data has intrigued me for a long time. Unfortunately, finding a way to convert analog sensor data to digital form can be a problem. Commercial A to D converters are too expensive for my limited budget, and the build-it-yourself kind are usually too complicated for my limited talents. Recently, however, I found a technique that makes A to D conversion simple and inexpensive. A converter costing less than $5 can provide good results and can be constructed in under an hour! The key is to make the PET do most of the work.

The process of converting a real-world value to a computer-usable form involves several steps. First, the value being measured must be changed into some electrical form. This electrical analog can be anything — voltage, resistance, capacitance — as long as it can be measured and made to change as the real-world value changes. The most commonly used sensors are of the voltage or resistance type. After the sensor makes the conversion to electrical form, the electrical value must be measured and put in digital form. Then the digital value usually must be mathematically converted into a standard scale of measurement. An A to D converter can do all or part of the work between sensor and the final result.

At one extreme, the converter can do all measurement and scaling and pass the result to the PET as a digital value with 4, 6, 8 or more bits. The circuitry required to do all of these functions and to accomplish any required handshaking can be quite complex.

Near the other extreme, the converter may only change the sensor value to a form that the PET can accept, and let the PET do the work of measurement and mathematical manipulation.

The latter method is far less expensive, since the only external circuitry is a single IC to convert the sensor changes to a PET-acceptable form. The technique I have tried uses changes in the sensor to control the frequency of a square wave oscillator or multivibrator. The square wave is then passed to one of the PET user port pins. By using the timer associated with the user port to measure the duration of a single pulse of the square wave, it is possible to arrive at a digital value which represents the original sensor measurement. This digital value can then be manipulated in BASIC to come up with a reading in standard measurement units. The process is quite simple and the results have been encouraging in terms of both resolution (the ability to distinguish between voltage or resistance values which are close together), and in consistency (the ability to get the same result on successive measurements of the same value). I have used two different circuits to convert sensor changes into frequency changes. They are the 74LS235 and the NE555. Each has its own advantages and disadvantages.

The 74LS235 is a voltage controlled oscillator (VCO), while the NE555 is a multivibrator. The 74LS235 provides much better consistency than the NE555 and it can be used with either voltage- or resistance-type sensors. The NE555 can only be used with the resistance type. On the other hand, the output (pulse width) of the NE555 is linear with respect to sensor resistance,

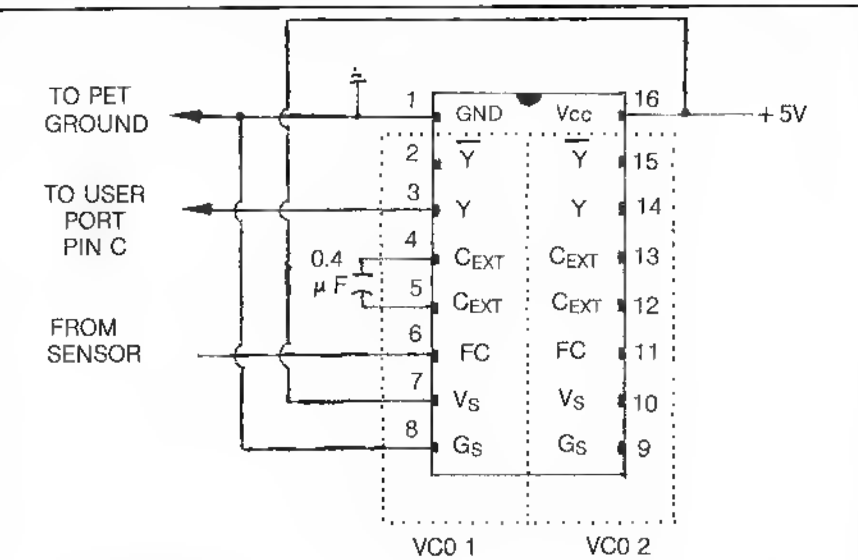


Figure 1: **Circuit diagram for the 74LS235.**

while the 74LS235's is not. With a linear output, the resolution of the circuit remains constant throughout its output range. Finally, the NE555 costs much less than the 74LS235.

The 74LS235

The 74LS235, made by Texas Instruments, contains two separate VCOs and costs about $4. The output of the circuit is a square wave with a frequency determined by the voltage applied to the Frequency Control (FC) pin, (pin 6 for VCO 1 or pin 11 for VCO 2). The circuit is capable of operating at frequencies from .12Hz to 30MHz. The actual operating range at any time is determined by a capacitor, C_{ext}, connected to pins 4 and 5 (or pins 12 and 13). For example, a C_{ext} of .4 μ f will produce an output of around 150Hz to 750Hz. Other pin connections are:

Gnd — Power supply ground
Y, $\overline{Y}$ — Output of VCO, alternate (inverse) output
V_s — Signal voltage, connect to V_{cc}
G_s — Signal ground, connect to Gnd
V_{cc} — +5V power supply

Figure 1 is a wiring diagram for measuring an external voltage with the 74LS235. A resistance can be measured in the same way by connecting the resistance between V_{cc} and the FC pin in place of the external voltage. The output curve of the VCO is shown in figure 2. The horizontal axis is labeled frequency but more correctly represents pulse duration, the inverse of frequency. The points on the curve are readings taken at .25V increments from .5V to 5V. The fact that the points are farther apart at lower voltages means that the circuit provides better resolution for lower sensor voltages (or higher sensor resistances).

A highly desirable characteristic of the 74LS235 is its consistency of output. If the circuit is calibrated and then turned off, it does not need to be recalibrated the next time it is turned on. Figure 3 illustrates this consistency. The points on the graph represent several trial runs made over a period of days. Actual voltage was measured by a voltmeter; test voltage by the PET/74LS235. Ideally, the actual and test voltages will be the same and all points will fall on the 45 degree diagonal. In fact, the largest deviation between actual and test values is .08V!

The NE555

When connected as shown in figure 4, the NE555 operates in the astable mode and produces a continuous square wave output. The shape of the square wave is determined by R_a and R_b. The "output low" portion of the wave is proportional to R_b. The "output high" portion is proportional to $R_a + R_b$. The sensor should go in the R_a location and R_b should be a resistor large enough (typically 10K) to insure that the output pulse stays low long enough for the measurement program to detect the change in state. The output frequency can be kept in a desired range by proper selection of C_{ext}. The relationship between C_{ext} and the output is that the output is designed to stay high for a time equal to $.69 \times C_{ext} \times (R_a + R_b)$. C_{ext} should be chosen for a high-pulse duration of about 1 millisecond. For example, if $R_a + R_b = 140K$ C_{ext} should be about .01 μ f. The pulse duration of the circuit is quite linear with respect to sensor resistance and it thus avoids the variable resolution problem of the 74LS235. On the other hand, it does not have the consistency of the 74LS235 and should be recalibrated with each use.

Figures 5a and 5b are oscilloscope traces which show how the square wave output of the NE555 changes as R_a and R_b are changed. An interesting point is that this relationship makes it possible to measure both R_a and R_b at the same time. R_b can be measured by timing the output low portion of the wave and R_a can then be measured by timing the output high portion and then subtracting the output low time from the output high time. This makes it possible to measure two sensors on a single user port pin. A joystick, for example, can be connected so that one potentiometer is used as R_a and the other for R_b on the same NE555. This is a distinct improvement over the use of the NE555 in the monostable mode which requires three pins. A suitable resistor must be placed in series with the R_b potentiometer so that the output low pulse does not get too short to detect.

The Measurement Program

Timer T1 is used to measure the output frequency of the 74LS235 or NE555. Here are the required steps:

1. Set timer T1 latch to the maximum value, decimal 255.

2. Wait until the circuit output is low.

3. When the circuit output goes high, start the count-down on T1.

4. When the output returns to low, stop timing. The final timer value subtracted from the original value is the duration of the output high pulse of the square wave.

You need to use assembly language programming to be able to sample the

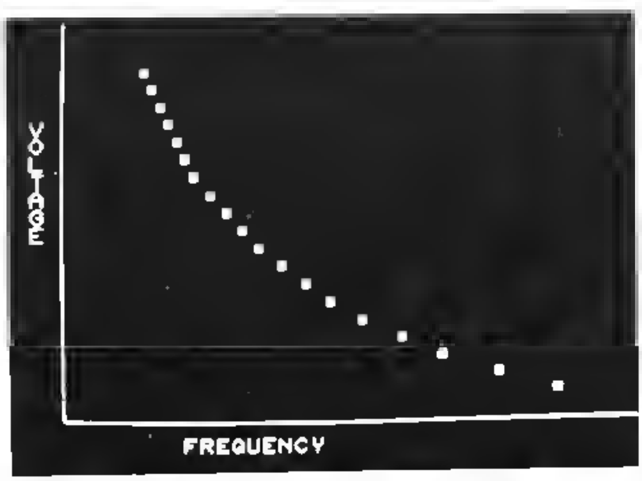


Figure 2: **Output curve of the 74LS235.**

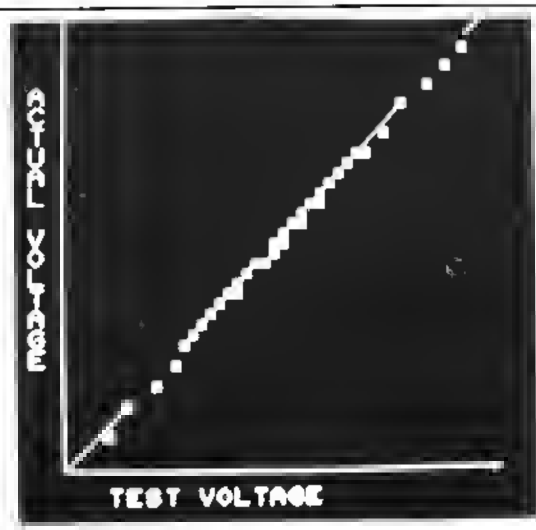


Figure 3: **Consistency test of the 74LS235.**

output rapidly enough for accurate timing; but even with assembly language the resolution capability of the process is limited by the fact thar it takes 7 microseconds to check the output, find out wherher it is high or low, and then check again. Therefore, pulse duration can only be measured in increments of 7 microseconds or more. If the pulse durarion is considerably longer than this 7 microsecond sampling time, resolution should not be a serious problem. For frequencies above abour 2KHz, however, it could be a major source of error.

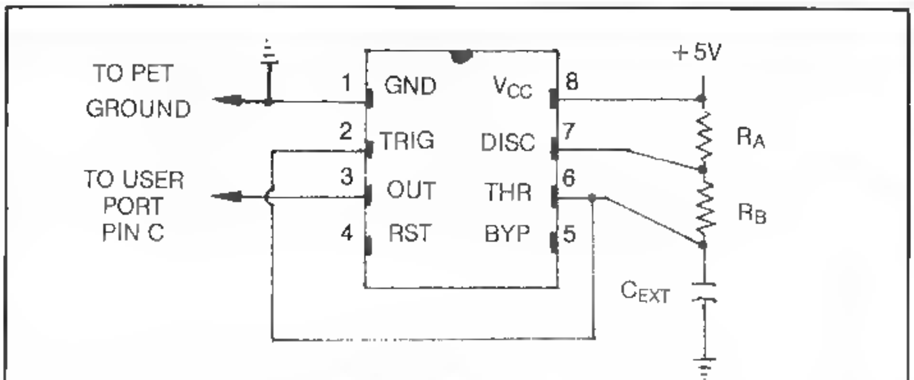


Figure 4: **Circuit diagram for the NE555.**

The accompanying BASIC program loads an assemhly language routine to measure output frequency. When the program is run, it first makes the top 256 bytes of RAM inaccessible to BASIC. It then loads the assembly language routine into thar space. Afrer the BASIC program has been run, another BASIC program may be loaded over it. The assembly language routine is entered with the statement SYS (7937). After the assembly language routine has been run, the final timer value is stored in bytes 8190 and 8191 (low order byte, high order byte). Since each rimer byre is counted down from 255, rhe pulse duration can be returned with the statemenr:

```
PRINT 255 – PEEK(8190) + 256*
(255 – PEEK (8191))
```

The program is written for the old ROM, 8K PET. For upgrade ROM, the top of memory address — bytes 134 and 135 in lines 100, 110, and 120, should be changed to 052 and 053. User port and timer addresses are rhe same for either ROM. For PETs with more than 8K RAM, the values in lines 292 and 294 should be changed to indicare the top two bytes of RAM. The SYS and PEEK commands should he changed to refer to the top of RAM minus 254 and the last two bytes of RAM, respectively. The program assumes the circuit output is connected to pin PA0 of the user port. A different pin can be used by changing line 320 of rhe program to replace 001 with 2^n for pin PAn. The program will rerurn a timer reading of zero if the timer limit of 65536 microseconds is exceeded.

In line 470, the program adjusrs the timer value to "correct" for the time previously consumed in checking its value. Jitter and other minor timing problems are corrected by waiting until

A B

Figure 5: **Oscilloscope traces showing relationship ot R_a, R_b and the output wavaform. At A, $R_a = R_b$ and the output is high twice as long as it is low. At B, $R_a = 0$ and the high pulse is tha sama length as the low pulse.**

three consecutive pulse measurements with the same value are obtained. If the circuit is very jittery it may take a while to get three consecutive identical readings. The problem becomes more severe with long pulse durations, i.e. frequencies below about 100Hz. Thus, resolution one way and jitter the other establish a desired operating range of 100Hz to 2000Hz.

The timer measurement can be reconverted to voltage or resistance value by use of either curve fitting or table look-up. With curve fitting, a series of measurements is fit to a mathematical equation. The equation will be of the form Y = A/X + B for tbe 74LS235 and tbe form Y = AX + B for the NE555. With the table look-up technique, a table of actual/test measurement pairs is kept in memory and an unknown test measurement is found by comparison with the stored values. Table look-up has the advantage of allowing simultaneous conversion of the resistance or voltage reading to the outside world value being measured.

Miscellaneous Notes

The ground connections shown in tbe circuit diagrams assume tbat an external power source is used. It is also possible to tap +5V from the PET second cassette interface. This source is commonly used although Commodore does not recommend it. If tbe PET power supply is used, the IC ground needs to be connected only to the PET ground.

A more complete discussion of tbe NE555 can be found in the *TTL Cookbook* by Don Lancaster (Howard W. Sams & Co.). I have not found a similar reference for tbe 74LS235.

```
100 SL=PEEK(135)-1
110 POKE135,SL
120 SL=256*SL+PEEK(134)+1
130 POKE59459,0
290 REM-------MAIN PROGRAM
292 DATA141,254,031:REM    STA $1FFE
294 DATA142,255,031:REM    STX $1FFF
296 DATA160,002    :REM    LDY #$02
300 DATA162,255    :REM    LDX #$FF
310 DATA142,068,232:REM    STX $E844
320 DATA169,001    :REM    LDA #$01
330 DATA044,079,232:REM    BIT $E84F
340 DATA208,251    :REM    BNE $FB
350 DATA044,079,232:REM    BIT $E84F
360 DATA240,251    :REM    BEQ $FB
370 DATA142,069,232:REM    STX $E845
380 DATA044,079,232:REM    BIT $E84F
390 DATA208,251    :REM    BNE $FB
400 DATA174,069,232:REM    LDX $E845
410 DATA169,064    :REM    LDA #$40
420 DATA044,077,232:REM    BIT $E84D
430 DATA240,006    :REM    BEQ $06
450 DATA162,000    :REM    LDX #$00
460 DATA169,000    :REM    LDA #$00
470 DATA240,006    :REM    BEQ $06
480 DATA173,068,232:REM    LDA $E844
490 DATA024        :REM    CLC
500 DATA105,013    :REM    ADC #$0D
510 DATA236,255,031:REM    CPX $1FFF
520 DATA208,196    :REM    BNE $C4
530 DATA205,254,031:REM    CMP $1FFE
540 DATA208,191    :REM    BNE $BF
550 DATA136        :REM    DEY
560 DATA208,196    :REM    BNE $C4
570 DATA096        :REM    RTS
590 DATA999
1000 REM------LOAD ASSEMBLY LANGUAGE
1010 READ A
1020 IF A > 255 THEN END
1030 POKE SL,A
1040 SL= SL+1
1050 GOTO 1010
```

Substitute Characters

By Loren Wright

The PET, unlike many home computers, can display 128 different characters on the screen at the same time. For many PET owners, this capability is adequate compensation for the lack of color or high-resolution graphics. For most serious applications, the convenience of manipulating screen information in the form of characters is very important.

The real bonus, though, is that there are actually two different 128-character sets in every PET, but only one of the two sets can be used at a time. One set has the full array of PET graphic characters (POKE 59468,12), while the other provides lower case at the expense of several graphic characters (POKE 594658,14). Many of us program primarily in one of these two sets. Business programmers need the lower case, and game programmers need the graphic characters. That other set remains unused most of the time.

What if you could change that dormant character set to any set of 128 characters you want? Let's say you want to type foreign language documents, complete with umlauts and cedilles. Your second character set could include these special characters. If you want to display calculus equations, it would help to have an integral or sigma character available. Fortunately substitute character ROMs are commercially available for foreign language and math applications. The ROM includes the standard character set along with the alternate math or foreign language set.

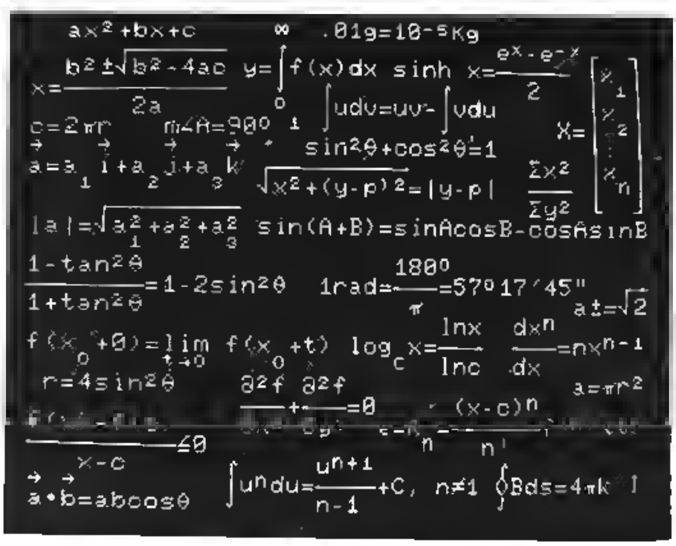
West River Electronics R & D (P.O. Box 605, Stony Brook, New York 11790) offers a math character set ROM and a foreign language ROM for \$60 each. These substitute ROMs directly replace the 24-pin character ROM. Only the lower case (POKE 59468,14) character set has been replaced; the graphics set has been retained in its entirety. With a Spacemaker you can switch back and forth between the standard ROM and either or both of the West River substitutes. (See figure 1 for sample screen displays using these characters.) West River Electronics now has printer (CBM 2022, 2023, 4022) ROMs available with these character sets to match the screen display.

This article is for those who have special needs in a character set. Much work is involved in planning your extra character set, coding it and programming an EPROM, so it is a good idea to know what you want before you start.

The idea of substituting character sets is not limited to the PET. The VIC has a configuration similar to the PET's, but it has the added capability of reading a character set programmed in RAM. OSI and PET/CBM machines normally read characters programmed in a mask-programmed ROM called the character generator. This ROM can be replaced with an erasable ROM (or EPROM) programmed exactly to your needs. The 2716 EPROM has dropped considerably in price in recent years, and EPROM programmers are now priced well within the reach of most professionals and many hobbyists.

The PET's character ROM can be substituted directly with a 2716. In

Figure 1: **Foreign language and math character sets for PET from West River Electronics. (Photos of PET screen.)**



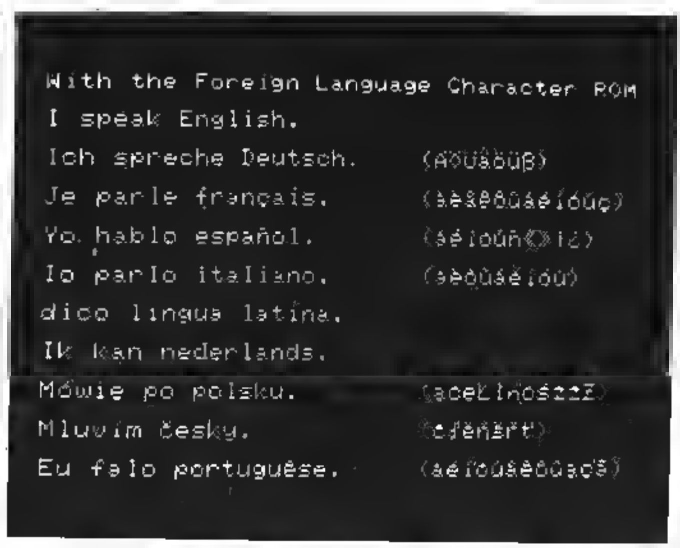
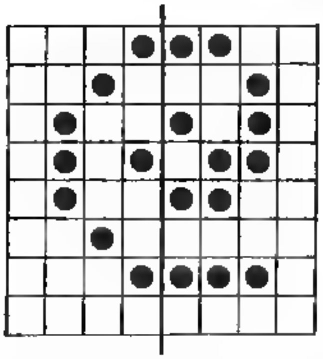


Figure 2: How characters are coded.



Binary	Hex
0001 1100	1C
0010 0010	22
0100 1010	4A
0101 0110	56
0100 1100	4C
0010 0000	20
0001 1110	1E
0000 0000	00

order to substitute a 2716 for an OSI character generator, though, you must pull two of the 2716's pins (18 and 20) out of the socket and ground them. Normally you would choose one of the standard sets for half of the EPROM and replace the other set with your own, custom-designed character set.

How the PET Displays Characters

The PET screen holds 1000 characters in a 25 × 40 array. Each character is actually an 8 × 8 array of little dots called pixels, which can be either on or off to comply with requirements of a particular character. The information regarding which dots are on and which are off for a particular character is stored in the character generator. Since the image on the screen deteriorates quickly, it is refreshed every 1/60 second by the PET.

Every character position on the screen has a corresponding memory location in RAM. These start at $8000 (32768), and end at $83D7 (33767). Each RAM location contains the PET/CBM screen code (not ASCII) corresponding to a character on the screen.

To demonstrate this character representation switch to the graphics character set (POKE 59468,12), move the cursor to the upper left hand corner, and type "ABC012". Move the cursor down one line and type "FORI=0TO5:? PEEK(32768 + I); : NEXT" and <RETURN>. This displays the screen codes for the six characters you typed on the first line. The digits' codes are the same as in ASCII, but the letters' codes start at 1 rather than 65. Another way to demonstrate the relationship of screen and the display is to enter the monitor and display memory starting at $8000. Changing the contents of these locations will make instant changes in characters on the screen. OSI machines have different screen RAM locations and a different (also non-ASCII) character coding scheme.

After the proper character is looked up in display RAM, the character generator is consulted for the appropriate pattern of pixels, and the electron beam is turned on or off appropriately on its way across the screen. Actually it's a little more complicated than that, since printing a line of characters requires eight passes of the beam. The pixel patterns for the first row of each of the 40 characters in the line are displayed on the first pass, followed by the second row's patterns, and ending with the eighth's. A character is not completed until the eighth pass. Meanwhile, the other characters on the line will get completed, too.

Each character in the character generator is coded by eight bytes: one for each row of pixels. Within the byte, a 1 indicates that pixel will be on, and a 0 indicates off. Figure 2 shows an '@' sign enlarged so that the pixel patterns can be seen. To the right of each row is the corresponding byte in the character generator, represented in both binary and hexadecimal. You should see the pattern clearly. Incidentally, '@' is the first character in the generator ROM, represented by the first eight bytes. The next eight bytes represent the letter 'A'.

Big Letters

Physics professors at University of California Berkeley lecture to undergraduate classes of 100 to 600 students. Many demonstrations, particularly in electricity and magnetism, require the display of various meter readings to the students. Projection analog meters, with their low accuracy and limited ranges, had been used for this purpose. Looking for a way to display readings of our new digital multimeter to a large class, I went to the department's electrical engineer. I expected to walk away with a bank of large LED digits, but ended up designing a character set for the PET!

Two of the five lecture rooms we used had large TV monitors, and we had a pair of portable ones. The idea was to use the PET to process the meter readings and display its screen on the lecture room monitors. However, even with these large monitors, standard size PET characters were unreadable more than a few feet away. We needed characters that were four or five times larger than the normal ones, and the only way to do it, short of serious hardware modification, was to use four standard characters to form one big character.

It is possible to design a set for this purpose with relatively few interchangeable component characters, but, as you can see, we ended up with something a lot better. Figure 3 shows the components of a big 'A', with a standard 'A' for comparison.

These characters cannot be conveniently used from the keyboard, since four different characters are required for each large character displayed. Also, extensive repositioning of the cursor is required. It is therefore desireable to write a display routine, preferably in machine language.

Figure 3: Comparison of standard and "big" letters.

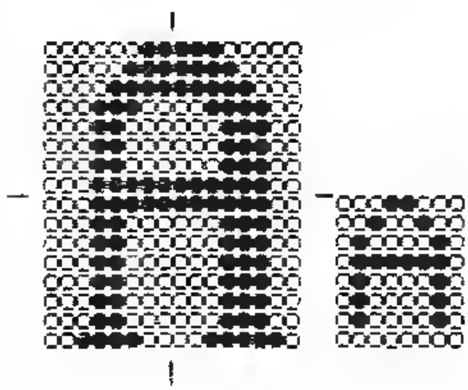


Figure 4: Sample of "Big Letter" characters on screen. Small 1 and 2 are standard characters.

The one I wrote searched specifically for the BASIC variables X, Y, and LS$ (for large string). X and Y determine the position from the upper left origin, and LS$ the desired output in large characters. All of the capital letters and numbers were available, as well as a few special purpose (physics) characters like mu, rho, omega, small m, n, and z, and small 0, 1, and 2. The latter were assigned convenient keyboard representations, such as shift 'O' for omega.

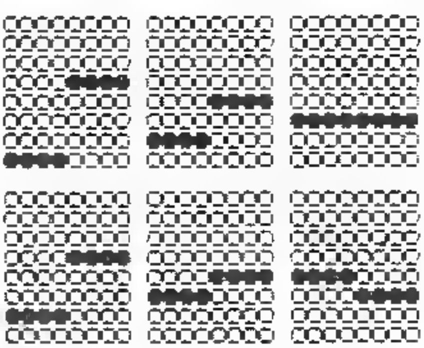
The machine language routine looked up the component characters in a table and displayed them with cursor controls properly inserted. Of course BASIC program listings are unreadable. That's why you keep the other character set in the ROM pretty standard!

High-Resolution Plotting

With this character set (we called it PPLOT), you can plot a single pixel anywhere on the screen. This is a little misleading, though, since only one dot can be displayed at a time in a given character space. It is high-resolution plotting, but of a rather low density. This set is most effective when used in the dynamic mode. That is, each dot is erased (if necessary) before the next is plotted, and it appears that a single point is moving across the screen. With a proper machine language driver, you can even do an imitation of an oscilloscope. We used it in the classroom for drawing circles, ellipses and electric potential lines.

Since only 64 characters are required to represent the 64 possible positions of a single dot within a character, all the letters and numbers can retain their original positions. It is important, however, to keep all 64 together in a logical order, so that the display software is easy to write. As a result, some of the characters used to display origins and axes had to displace some of the punctuation characters.

Figure 5: **Sample of PPLOT characters.**



200 × 80 Resolution Plotting

In the normal PET graphics character set, all 16 horizontal and vertical line characters are available. This means that you can plot with the horizontal lines and achieve 40 × 200 resolution (8 lines in each of 25 characters). This character set increases the horizontal resolution from 40 to 80 by using, as characters, all the possible combinations of horizontal lines which are half the width of a character. There are 81 of them, including the 17 where half or all of a character is blank. I already had a routine to plot with the full-width horizontal lines which I called 'HPLOT', so a logical name for this character set was 'FPLOT' since the lines are only 'F' a character in width! The appearance of a plotted curve is considerably better than the equivalent HPLOT version, and much better than the QPLOT version (quarter boxes).

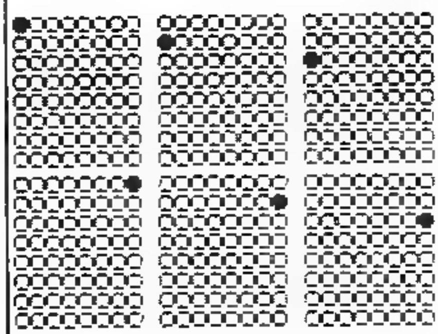
Again, it is important that all 81 characters remain together and in a logical order. This does foul up the appearance of BASIC listings somewhat, since allocating a continuous run of 81 characters without touching the letters or numbers is impossible. Programming in this character set is a bit more of a challenge, since you must look ahead to find out what's already in the position you want to plot. I wrote a machine language routine to handle this, with the BASIC variables X and Y as input.

More Ideas

One project (never completed) in the electronics shop of University of California Berkeley physics department, involved using the PET to lay out printed circuit boards on the PET screen. The board could be larger than the screen, and it could be flipped over. When the layout was completed, it was output to a digital plotter, which drew the pattern on paper or directly onto the PC board. A very large special character set was required for this.

To get 256 different characters on the screen, a hardware change was made which resulted in the sacrifice of reverse field. Many people will want to add characters for tanks, robots, or space invaders. Musical notes, drafting symbols, and many other special characters are possible.

Figure 6: **Sample of FPLOT characters.**



Programming Considerations

If you're going to use your character set from the keyboard, give a lot of thought to which keys will be used. There's no point in having characters available which you can't find! Remember that the space is a character, even if it has no pixels on.

Graph paper is very handy for designing the characters. Or, you could use a modification of Roger Crites' character dictionary program (MICRO 37:11). The printer characters he used are 5 × 7 and were stored in columns rather than rows.

Finally, I can't overemphasize the value of good, continuously updated documentation.

Acknowledgements

Several current and former members of the U.C. physics department contributed significantly to this project. Kim Rubin was responsible for the big letters concept and did most of the design of those characters. Prof. Leroy Kerth came up with the PPLOT character set. John Davis and John Girard were responsible for considerable engineering and technical support.

Loren Wright is a member of the MICRO staff, serving not only as "PET Specialist," but also in a number of other roles. Before coming to MICRO, he served as instructional technician for the physics department at the University of California, Berkeley. He is also a trained marine biologist and a railroad enthusiast.

MICRO

The 6809 and the S-50 Bus

Editorial note: The author presents a brief history of the S-50 bus and a list of supporting manufacturers. Those listed under "Major Hardware Manufacturers" and "Support Manufacturers" are only those which relate to the S-50 bus. Other bus systems are supported by Motorola (the manufacturer of the 6800 and 6809) and by Radio Shack in its TRS-80 Color Computer. The Computerist supports the KIM-4 bus, a standard in the 6502 world, and there are numerous manufacturers supporting the S-100 bus system. The software available from the manufacturers listed is relatively bus-independent.

Dale L. Puckett
14753 Endsley Turn
Woodbridge, Virginia 22193

I was happy to see MICRO Editor Robert Tripp decide to publish information about the Motorola 6809. I am also glad that he has given me the opportunity to share information about 6809 systems with MICRO readers.

I switched over to the 6809 from the 6800 about 18 months ago and I now find it very hard to go back. Don't get me wrong, the 6800 is not difficult to program, it's just that the 6809 is much easier to work with.

When I started using the 6809, my programs became about 30 percent shorter and ran about 30 to 40 percent faster. I wound up with a much better product and had less work to do. Features like the auto-increment instructions make routine programming chores a breeze. And, position-independent code is so easy to generate that you really don't have an excuse for not writing state-of-the-art programs.

This article will deal with equipment and software that is already available in the 6809 world. Hopefully, it will help you decide to start using this excellent processor right away.

Dale Puckett is a professional writer who contributes regularly to *Info World* and *68 Micro Journal*. He owns 6800 and 6809 systems, and is the author of several software packages available from Frank Hogg Labs.

Major Hardware Manufacturers	Products
Southwest Technical Products 219 West Rhapsody San Antonio, TX 78216 512-344-0241	complete computer systems disk systems terminals
GIMIX, Inc. 1337 West 37th Place Chicago, IL 60609 312-927-5510	complete computer systems disk systems memory mapped video boards
Midwest Scientific Instruments 220 W. Cedar Olathe, KS 66061 913-764-3273	complete computer systems disk systems
Smoke Signal Broadcasting 31336 Via Colinas Westlake Village, CA 91361 213-889-9340	complete computer systems
Percom Data Company, Inc. 211 North Kirby Garland, TX 75042 214-272-3421	disk systems various CPU and memory mapped video cards

Support Manufacturers	
Thomas Instrumentation 168 Eighth Street Avalon, NJ 08202 609-967-4280	modem card motherboard and memory cards memory mapped video board
Boaz Co. Box 18081 San Jose, CA 95158	dynamic memory card

(continued)

The Standard Bus

We should look first at the roots of the 6809. To do this, we must go back to 1974 and meet a San Antonio businessman named Dan Meyer. During that year his company, Southwest Technical Products Company (SWTPC) started building an inexpensive microcomputer based on a design supplied by Motorola. It used a 6800 microprocessor and featured what the company called the S-50 bus.

Over the past seven years that bus has become a standard that is supported by a dozen major manufacturers and software houses. A few dozen more companies are on the bus now, supplying peripherals and writing applications software.

There is a good reason for the popularity of this bus, and it is one you should consider when you think about moving up to the 6809. Any board designed for the standard S-50 motherboard will run in a box built by every other manufacturer. Until very recently, even the I/O addresses and monitor jump tables were the same on all the machines. Still, there is only one maverick. This is important because it allows software written for one brand of machine to run on the others with no patching.

The S-50 manufacturers really started getting it together at several computer shows in 1980. It began at the West Coast Computer Faire when they adopted the slogan, "Take the S-50 Bus — the Choice is Great!" Then at Philadelphia in August, fifteen S-50 exhibitors got together and reserved a large area of the show exclusively for S-50 products. They also set up a mini-auditorium and presented speakers describing the features of all the new hardware and software. Everyone in the crowd got his or her questions answered.

The Philadelphia gathering was the idea of Richard Don, president of Gimix, Inc., in Chicago. He says it best.

> People assume that the S-100 bus is better because there are more manufacturers on it. But they don't realize that many of the companies define some of the bus lines differently. This creates a number of different S-100 buses. There is only one S-50 bus.

Don also noted that the S-50 bus has better development software and its operating systems are superior to those available on other buses. A quick trip around the exhibit area in Philadelphia proved his point. Applications packages were already appearing at several booths — a direct result of the ease of programming the 6809.

The 6800 chip got a bad name early in the competitive game because of its first BASIC interpreter. It was an excellent language and even had ten-digit precision while everyone else was settling for six. It was cheap too, (only $10 for the 8K version), but it was slow.

Because this BASIC was slow everyone thought, (or at least those who lived to program in BASIC), that the 6800 was a slow processor. Those who were faithful to the S-50 bus got in the last word though when Technical Systems Consultants (TSC) released their 6800 BASIC. It turned out to be the fastest BASIC on any 8-bit processor. The 6809 version released a year later was even faster (by 30 percent).

When the 6809 appeared, . things seemed to come together for the S-50 crowd. The processor is fast, powerful and lends itself to multiprogramming and multitasking. Yet, it is still simple to use and to understand.

There are now S-50 systems with as little as 4K of memory and others with as much as 384K. You can find hard storage units holding several dozen megabytes and systems that will support a large number of users.

S-50 systems have always had a ROM monitor and have worked with an external CRT terminal. Now, most manufacturers also sell video boards, and you can put together a system which will fit in one box. At the same time, the system is expandable. As long as you have slots left on your motherboard (some have as many as 15 slots) you can add memory or peripherals to your heart's content.

At this point, a survey of the major hardware manufacturers of the S-50 bus is in order. SWTPC is still in business. They sell 6809 computers ranging from kits with 8K of memory, to assembled and tested units containing 256K. They also sell 5¼-inch disk systems that will store from 720K to 1.4 million bytes on two drives, a dual drive 8-inch system featuring a direct memory access controller, and a Winchester-type hard disk. SWTPC's 8209 and 8212 terminals have become standards in the 68XX world.

The Cadillac of the 6809 world is the system sold by Gimix, Inc., in Chicago. Richard Don has nicknamed his mainframe "The Classy Chassis" and not without good reason. It has a 25 amp constant voltage power supply for full

protection from brownouts, a 15-slot motherboard, and is packaged in a heavy-duty aluminum cabinet. Every card in the GIMIX system is completely decoded and may be addressed quickly with built-in dip switches.

The GIMIX memory boards are fully decoded to use the four extended address lines on the S-50C bus. This allows you to address them anywhere in the first one million bytes of memory. Gold connectors are used on every board, and every card is guaranteed to run at 2 Mhz. The GIMIX 80 × 24 video board is the most versatile on the market, and when used with a good word processing package like Bob Bundy's Stylograph, it makes writing very easy. Don's latest addition is a 64K memory card which uses CMOS RAM. It comes complete with a battery back-up and will hold your data for several months.

Other major hardware manufacturers of the S-50 bus include Smoke Signal Broadcasting in Westlake Village, California, Midwest Scientific in Olathe, Kansas and Percom Data Company in Garland, Texas.

Midwest Scientific was one of the first companies to introduce a disk system for the bus. They entered the field in 1976 with an 8-inch dual drive system and have been adding to their product line ever since. They provide full software support and sell a line of application packages for businesses.

The popular BFD-68 disk system was the catalyst that got Smoke Signal into the S-50 business. A few years later this triple drive system had evolved into the Chieftain, a complete microcomputer with 32K of user memory and built-in disk drives. The latest entries from Smoke Signal include a 6809 CPU card and a double-density disk controller card that will let you store 366K bytes on one 5-inch floppy disk. Smoke Signal is the only major supplier that has moved its I/O slots from the set of standard addresses used by all other companies.

Percom's main business is in the 5-and 8-inch disk market. But the company does supply a motherboard and both 6800 and 6809 CPU cards. They also sell a video display board called the Electric Window.

There are a number of smaller companies around that sell motherboards and inexpensive components. Two of them, Thomas Instrumentation in Avalon, NJ, and "febe" in York, PA, can supply anything you need to build a custom system.

Support Manufacturers	Products
Febe Group 51 Hamilton Ave. York, PA 17404 717-854-0481	motherboard with power supply and case
Star-Kits P.O. Box 209 Mt. Kisco, NY 10549	single board 6802 computer software
Microdyne P.O. Box 1707 Greenville, MS 38701 601-335-9321	active bus terminator
JPC Products Co. 12021 Palsano Ct. Albuquerque, NM 87112 505-294-4623	tape units cassette interface analog/digital converter board
Hazelwood Computer Systems 7413 North Lindbergh Hazelwood, MO 63042 314-837-3466	high resolution video graphics board
Wave Mate, Inc. 18005 Adria Maru Lane Carson, CA 90746 213-532-4532	Stand-alone 6800 computer with video terminal and disk drives
F & D Associates 1210 Todd Road New Plymouth, OH 45654 614-592-5721	single board computers memory mapped video boards and various CPU cards
Compuware Corporation P.O. Box 2710 Cherry Hill, NJ 08003 609-428-2309	hardware calendar and clock with interval timer
Digital Research Computers P.O. Box 401565 Garland, TX 75040 214-271-3538	RAM memory
Digital Service and Design P.O. Box 741 Newark, OH 43055 614-366-6314	memory cards
Newtech Computer Systems 230 Clinton Street Brooklyn, NY 11201	music and analog/digital cards
Peripheral Technology Assoc. 290 Lamplighter Lane Marietta, GA 30067	disk controllers
Optimal Technology, Inc. Blue Wood 127 Earlysville, VA 22936 804-973-5482	EPROM Programmer
Southeastern Microsystems P.O. Box 293 Conveyers, GA 30207 404-922-1620	dynamic RAM cards

Tbomas has an impressive new card on the market that will give you an auto answer/originate modem using the Bell 103 standard, a real time clock, two serial ports, two parallel ports, and automatic telephone answering. The software supplied will also give you automatic dial pulse or touch-tone dialing.

Software

No matter how good the hardware, you can't do anything without the software to run it. This, too, is a big plus for the 6809 user. Just as the manufacturers chose to stick witb the same bus definition, most of the software houses stuck with one disk operating system.

FLEX, written by Technical Systems Consultants, Inc., in Lafayette, Indiana, became the standard on both the 6800 and 6809 systems and is wholeheartedly supported by all but one hardware manufacturer. This common operating system has given independent software houses a chance to write software that will run on any manufacturer's equipment, including the Motorola Exorcisor.

TSC has just released a new generation of operating system for use on 6809 and 68000 systems. Called UniFlex, it is a multiuser, multitasking system similar to the Bell System's UNIX. It uses a hierarchical file system and device-independent I/O. Since it requires a minimum of 96K of memory, TSC continues to support FLEX for single user systems.

Major Software Manufacturers	Products
AAA Chicago Computer Center 120 Chestnut Lane Wheeling, IL 60090 312-459-0450	text processor plus some hardware and mailing list program
Roberts Control Equipment 3640 Western Road, Unit 3 Weston, Ontario M9L 1W2 Canada 716-631-8178	dynamic RAM cards
Blue Hat Software Co. Box 4127 Flint, MI 48504 313-738-2863	DOS for Percom disks
Technical Systems Consultants Box 2570 West Lafayette, IN 47906 317-463-2502	disk operating systems BASIC, Pascal, C, and application programs
Microware Systems Corp. 5835 Grand Avenue Des Moines, IA 50304 515-279-8844	OS-9 disk operating system BASIC09, Pascal and various systems programs
Frank Hogg Laboratory 130 Midtown Plaza Syracuse, NY 13210 315-474-7856	FORTH, job control program, data base management program, remote terminal program, English text analysis and artificial intelligence demo
Alford and Associates P.O. Box 6743 Richmod, VA 23230 804-320-6722	screen editor, real time TREK

(continued)

On the 6809, FLEX may be getting some very stiff competition from a new operating system written by Microware Systems Corporation, of Des Moines, Iowa. OS-9 also uses UNIX-like files and is a real-time, multitasking hardware-dependent system. It comes in two versions: Level One, which requires 32K to 56K of memory, and Level Two, which is designed to use up to a million bytes of memory on a system with memory management hardware. Both OS-9 systems give you complete timesharing capability and a device-independent I/O that can handle almost any number and combination of devices, ranging from 5¼-inch floppies to Winchester hard disks, and from standard serial and parallel ports to memory-mapped video displays.

OS-9 was written by Microware, under a contract with Motorola, to provide an operating system to support the company's BASIC09. This compiler/interpreter will run standard BASIC programs or slightly modified Pascal code. It contains all the control statements needed for structured programming and gives you defined data types and complex data structures.

Frank Hogg Laboratory in Syracuse, New York, has become the major international distributor of applications and systems software for the 6809. Hogg presently supplies all programs for the FLEX system, but will soon be adding OS-9 formatted disks.

Hogg also sells a FORTH interpreter that is totally compatible with FLEX systems. It even uses standard FLEX formatted files instead of the hybrid FORTH file design. It is very fast, contains an excellent full screen editor, and is shipped with a 400-page manual that will soon become the basis of a book about FORTH released by a major publisher.

Another Hogg program that is very popular is DATAMAN, a complete Database Management System. It is one of the most complete packages on the market and is made up of 16 menu-driven programs. It allows, among other things, any number of fields and any number of bytes in a record. Many DBM programs limit you to 254 bytes or one sector per record. A companion package called DATARAND converts DATAMAN sequential files into fast random access files which use hashed keys.

Hogg also features a Job Control Program that lets you automate your S-50 bus computer's operation, an intelligent

Software Manufacturers	**Products**
Computerware Software Services 1512 Encinitas Blvd. Encinitas, CA 92024 404-483-1717	operating systems BASICs
Talbot Microsystems 5030 Kensington Way Riverside, CA 92507 714-781-0464	tFORTH firmFORTH
Tallgrass Technologies Corp. 7623 West 86th Street Overland Park, KS 66212 913-381-5588	complete UCSD Pascal system
Dugger's Growing Systems P.O. Box 305 Solana Beach, CA 92075	small C compiler
MicroPower 1418 West Thorndale Chicago, IL 60660 312-989-8585	Juggler, a graphics game
Micropi 2445 North Nugent Lummi Island, WA 98262 206-734-8248	PILOT, and BLITZ screen editor
Micro Works P.O. Box 1110 Del Mar, CA 92014 714-942-2400	editor debugger, monitor and disassembler for TRS-80 color computer
Washington Computer Services 3028 Silvern Lane Bellingham, WA 98225 206-734-8248	RMS data base management system
Hemenway Associates, Inc. 101 Tremont St. Boston, MA 02108 617-426-1931	systems software
HHH Enterprises Box 493 Laurel, MD 20820 301-953-1155	SPIRIT, a FORTH-like language and various hardware systems
Mark Data Products 23802 Barquilla Mission Viejo, CA 92691 714-768-1551	various adventures
Omegasoft P.O. Box 70265 Sunnyvale, CA 94086 408-733-6979	Pascal compiler
Programma International 3400 Wilshire Blvd. Los Angeles, CA 90010	SPL/M compiler and PIE editor
Software Dynamics 2111 West Crescent Anaheim, CA 92801	disk operating system and BASIC compiler
Application Services Company P.O. Box 12227 Wichita, KS 67277	Adventure for 6800 and 6809 (the original version)

terminal program, a clever program that analyzes your English text and an interesting artificial intelligence demonstration program called ESTHER wbicb will amaze your friends for hours. Hogg is adding applications programs almost monthly, and plans to release a spelling check program this fall.

Other Systems

And, if a truly standard bus isn't enough to lure you into tihe 6809 world, the big names are coming. Sony is said to be working on a 6809 computer, and Canon is reported to be building two models, one designed to compete with Tandy's TRS-80 Color Computer. Apple is rumored to be working on a 68000 design, and Commodore has announced the Super PET (or Micro Mainframe) which adds a 6809 card to tbe 6502-based machine.

A listing of all the major manufacturers and software houses is provided here to give you a head start in your search for a 6809 system. I have listed each firm's major product and, where available, have provided tbe telephone numbers.

Software Manufacturers	Products
Sonex Systems Box 238 Williamsville, NY 14221 716-634-2466	Stylograph interactive word processing system
Universal Data Researcb, Inc. 2457 Wehrle Drive Buffalo, New York 14221 716-631-3011	Data Base Managcr
Westchester Applied Business Systems P.O. Box 187 Briarcliff Manor, NY 10501 914-941-3552	Data Base Manager
Computer Systems Center 13461 Olive Blvd. Chesterfield, MO 63017 314-576-5020	disassembler for 6800 and 6809
Dynasoft Systems Ltd. P.O. Box 51 Windsor Jct. Nova Scotia, Canada B0N 2V0 902-861-2202	tiny Pascal
Lucidata, Ltd. P.O. Box 128 Cambridge CB2 5EZ England	Pascal compilcr for 6800/6809

MICRO

Handling Analog Signals with a Micro

Problems of handling analog signals are discussed, and two techniques are presented.

Richard Soltero
Ciba-Geigy Corp.
556 Morris Ave.
Summit, NJ 07901

Arthur Poulos
Rutgers University
Newark, NJ 07102

```
                    ;*******************************************
                    ;
                    ;          ASSEMBLY ROUTINE FOR AN ICL7109 ADC
                    ;          ATTACHED TO A SYM-1 MICROPROCESSOR
                    ;
                    ;*******************************************
                    ;
                    .BA $B10
                    .OS
                    .LS
                   TEMP .DE $B0F     ;STORAGE FOR HI BYTE
                   LAD  .DE $F000    ;LOW ORDER BYTE
                   HAD  .DE $F001    ;HI ORDER BYTE
                    ;
0B10- AD 01 F0      LDA HAD          ;LOAD HI BYTE
0B13- 29 20         AND #$20         ;MASK FOR POLARITY BIT
0B15- F0 0D         BEQ NEG          ;IF POLARITY IS NEGATIVE,
                                     ; THEN 2'S COMPLIMENT IS NEEDED
0B17- AD 01 F0      LDA HAD          ;LOAD HI BYTE
0B1A- 29 BF         AND #$BF         ;MASK HIGH NIBBLE  + 4
0B1C- AC 00 F0      LDY LAD          ;LOAD LOW BYTE FOR TRANSFER
                                      TO BASIC
0B1F- 4C 4C D1      JMP $D14C        ;RETURN TO BASIC WITH A
                                      POSITIVE NUMBER
0B22- 18           NEG CLC
0B23- AD 01 F0      LDA HAD          ;LOAD HI BYTE
0B26- 8D 0F 0B      STA TEMP
0B29- AD 00 F0      LDA LAD          ;LOAD LOW BYTE
0B2C- 69 01         ADC #01          ;ADD 1
0B2E- 90 03         BCC CHKC         ;IF CARRY CLEAR,
                                      DON'T ADD 1 TO HI BYTE
0B30- EE 0F 0B      INC TEMP         ;ADD 1 TO HI BYTE
0B33- 49 FF        CHKC EOR #$FF     ;2'S COMPLIMENT ON LOW BYTE
0B35- A8            TAY              ;LOAD LOW BYTE FOR TRANSFER
                                      TO BASIC
0B36- AD 0F 0B      LDA TEMP         ;LOAD HI BYTE
0B39- 29 8F         AND #$8F         ;MASK HIGH NIBBLE + 4
0B3B- 49 FF         EOR #$FF         ;2'S COMPLIMENT ON HI BYTE
0B3D- 4C 4C D1      JMP $D14C        ;RETURN TO BASIC
                                      WITH A NEGATIVE NUMBER
                    .EN
```

Establishing a dialogue with a 6502 microprocessor is an important process which is facilitated by numerous varieties of terminals and keyboards. Some microcomputers, such as the AIM 65, support their own alphanumeric keyboard and a printer or display. Other micros, like the KIM and SYM, have an on-board hex keypad and LED display. By far the most popular, however, are CRT's and hard copy terminals which can communicate over RS-232 or 20mA current loop hook-ups.

These communication links can be used to write programs with the micro or to read the contents of various memory locations. When the microprocessor is used to talk directly to a data collecting instrument, the real potential of the small 6502-based systems is demonstrated. A micro such as the SYM can be set up in a wide variety of configurations to do any number of data transfer tasks. This is due to its 63 programmable I/O lines, eight control lines and five internal timers. This versatility also accounts for a degree of complexity in solving a communication problem. This article steps through several methods of getting alien information into a micro.

If the task is to bring in data from a measuring device such as a thermometer or a voltmeter, then several steps are involved. Since we are living in an analog world and wish to interface to a digital world, the first step is to put an analog to digital convertor (ADC) between the device and the micro. Once the ADC is in place, then the microprocessor must be somehow alerted to its presence so it can interact with it to bring the data into memory. After this, the programmer can write a program in a high level language like BASIC to crunch the data and print out tables or reports.

An example of setting up an ADC was presented by J.C. Williams in MICRO (12:25). He used a 100 microsec 16-channel ADC tied directly to the data bus of the 6502. Another example of this process will be described later in this article using a single channel 12-bit ADC tied to the data and address buses of a SYM. Dr. Marvin DeJong used a 6530 PIA to interface an 8-bit 75703 ADC to a KIM (MICRO 15:40).

Not all data must come from an ADC though. Dr. DeJong (MICRO

24:19) showed how a temperature could be measured and the data transferred through a 6522 VIA. In another article (MICRO 27:68), a method was described using the 6522 VIA to input binary coded decimal (BCD) digits from the digital panel meters of a spectrophotometer.

Not all ADC's can be used for all applications, therefore this article will show a way to deal witb slow analog signals and very fast analog signals. An instrument or measuring device puts out a voltage level which is proportional to a property being measured (i.e. air temperature, wind velocity, etc.). If the property does not change appreciably over a 1/10 of a second period, then an integrating ADC like the ICL 7109 can easily be used. If tbe property changes in less than a microsecond, then a very fast ADC is required. A fast 8-bit ADC is packaged inside a commercially available unit called the Biomation Waveform recorder, Model 8100. This unit will record the digitized information in 2K of RAM and then the entire block of memory can be read by a micro using the handshake routine described here.

One of the problems commonly faced in an analytical laboratory is to interface analog instruments with a computer. Ideally, a simple ADC with sufficient versatility and resolution for handling a variety of applications is part of the answer. The rest of the answer is a computer that can effectively provide the communication between the ADC, the CPU, and the operator. We have solved this dilemma with a few relatively inexpensive components. A SYM-1 with 4K RAM and BASIC ROM is used in conjunction witb a T.I. Silent 700 terminal for operator-computer communication. An Intersil 7109 ADC is used for communication between our instruments and the 6502 MPU.

The Intersil ICL7109 is a single channel ± 12-bit integrating ADC with polarity and overrange signals. It is tied onto the data bus with tristate logic so that either the low order bits or high order bits (+ polarity and overrange) can be selectively read on the same eight bus lines. The unit has latched outputs and was set in a free running mode where it will do 30 conversions a second. By just reading two addresses (at $F000 and $F001) the low and high order bits can be transferred into another memory location. An LS138 is used to decode thc address of thc low and bigh order bits. The 7109 needs a clock signal for its operation. This is provided by bringing in phase 2 of tbc SYM's clock signal and dividing it by 4 in a 74LS93.

```
                  ;*********************************************
                  ;
                  ;     ASSEMBLY ROUTINE FOR TRANSFER OF DATA FROM
                  ;     BIOMATION MODEL 8100 TO ROCKWELL AIM-65
                  ;
                  ;*********************************************
                  ;
                  ;
                 DRB .DE $A00      ;DATA REGISTER B
                 DRAH .DE $A001    ;DATA REGISTER A WITH HANDSHAKE
                 DDRB .DE $A002    ;DATA DIRECTION REGISTER B
                 DDRA .DE $A003    ;DATA DIRECTION REGISTER A
                 ACR .DE $A00B     ;AUXILIARY CONTROL REGISTER
                 PCR .DE $A00C     ;PERIPHERAL CONTROL REGISTER
                 IFR .DE $A00D     ;INTERUPT FLAG REGISTER
                 IER .DE $A00E     ;INTERUPT INABLE REGISTER
                 TABLE .DE $0F00   ;DATA TABLE
                 DSTRT .DE $90     ;DATA TAKEN CHECK
                  ;
                  .BA $0E90
                  ;*********  REGISTER INITIALIZATION  *********
                  .LS
0E90- A9 0B       LDA #$0B
0E92- 8D 0C A0    STA PCR          ;CA1 ACTIVE ON POSITIVE TRANSITION
                                   ;CB1 ACTIVE ON NEGATIVE TRANSITION
                                   ;CA2 IN PULSE OUTPUT MODE
0E95- 8D 90 00    STA DSTRT        ;CLEAR DATA TAKEN FLAG
0E98- A9 00       LDA #0
0E9A- 8D 03 A0    STA DDRA         ;MAKE PORT A INPUTS
0E9D- 8D 0E A0    STA IER          ;DISABLE ALL INTERUPTS
0EA0- 8D 02 A0    STA DDRB         ;MAKE PORT B INPUTS
0EA3- 60 .        RTS              ;RETURN TO BASIC
                  ;
                  .BA $0EB0
                  ;********  DATA TRANSFER SECTION  ***********
0EB0- A9 03       LDA #$03
0EB2- 8D 90 00    STA DSTRT        ;SET DATA TAKEN FLAG
0EB5- A9 0B       LDA #$0B
0EB7- 8D 0C A0    STA PCR          ;RELOAD PERIPHERAL CONTROL
                                    REGISTER
0EBA- AD 00 0A    LDA DRB          ;CLEAR CB1 FLAG IN IFR
0EBD- AD 0C A0    LDA PCR
0EC0- 09 C0       ORA #$C0
0EC2- 8D 0C A0    STA PCR          ;HOLD CB2 (OPT) LOW
0EC5- A2 00       LDX #0
0EC7- A0 08      WDSTR LDY #08     ;LOAD WORD COUNTERS
0EC9- AD 0D A0 NXTWRD LDA IFR
0ECC- 29 02       AND #02          ;CHECK IF CA1 (FLAG) IS HIGH
0ECE- F0 F9       BEQ NXTWRD       ;IF NOT CHECK AGAIN
0ED0- AD 01 A0    LDA DRAH         ;READ PORT A
                                   ;CLEAR CA1 FLAG
                                   ;SEND PULSE ON CA2
0ED3- 88          DEY              ;DECREMENT WORD COUNTER
0ED4- D0 F3       BNE NXTWRD       ;IF NOT 0, GET NEXT WORD
0ED6- 9D 00 0F    STA TABLE,X      ;STORE EIGHTH WORD IN TABLE
0ED9- E8          INX
0EDA- F0 07       BEQ OFF          ;AFTER 255 STORES, GO TO OFF
0EDC- AD 00 0A    LDA DRB          ;CHECK OFF FLAG
0EDF- 29 20       AND #$20
0EE1- D0 E4       BNE WDSTR        ;IF NOT 0, GET NEXT WORD
0EE3- AD 0C A0 OFF LDA PCR
0EE6- 09 20       ORA #$20
0EE8- 8D 0C A0    STA PCR          ;HOLD CBS (OPT) HIGH
0EEB- A9 02       LDA #02
0EED- 8D 90 00    STA DSTRT        ;RESET DATA FLAG
0EF0- 60          RTS              ;RETURN TO BASIC
                  .EN
```

The 12-bit word that is generated by this ADC allows a I in 4098 resolution as compared to an 8-bit ADC which allows only a I in 255 resolution. Voltage inputs, in this setup, can range between +5 and −5 volts. A schematic of the circuit shows how to hook up the address and data bus lines through the expansion (E) connector of the SYM.

The address lines are decoded in this circuit to put the data in high memory of the 4K version (at $F000 and $F001). The assembly program also resides in high memory ($810-$840) so that most of the RAM is free for a user-oriented BASIC program.

The assembly listing shows how the two address locations are manipulated

so that the polarity and overrange functions can be isolated from the low nibble of the high byte.

If the polarity bit is set, then the program returns a positive number to BASIC through the USR entry routine. If the polarity bit is low, then the assembly program branches to a routine where the high and low order bytes undergo a 2's complement. This allows a negative number to be returned to BASIC through the USR function.

The BASIC program listed below is all that is necessary once the connections have been made and an analog signal has been attached to the input of the ADC.

```
10 INPUT A
20 IF A #1 THEN 50
30 PRINT USR (2064,00)
40 GO TO 10
50 END
```

Each time a 1 is input in response to the question mark, the digital value of the ADC will be printed. Since the ADC converts continuously in the free running mode (about 30 times a second), all the programmer has to do is write a BASIC program which will invoke the USR function any time he wants to read a value of the analog signal.

The BASIC program can be designed to provide all of the delays, loops or massaging that the programmer may desire. This concept allows users who have had little or no hardware or software experience to control the routines and data in a more understandable high level language.

For handling fast analog signals, a waveform recorder like the Biomation Model 8100 is more suitable. The problem here is to transfer data from the recorder to a micro like the Rockwell's AIM 65. In this application, we have used a 4K version of the AIM 65 with BASIC ROM installed to transfer and process data from a Biomation. The Biomation will sequentially dump each of the 8-bit words in its memory onto eight pins of its output connector. These eight lines were connected to the eight pins of port A of the AIM's 6522 VIA. The four control lines of the VIA were also connected for control of other functions on the Biomation. The operation of the VIA was controlled by setting the correct bits in several of the registers associated with the VIA.

Table 1 shows the connections between the Biomation's output connector (J-10) and the application (J-1) edge connector of the AIM 65.

```
10 REM BASIC PROGRAM FOR CONTROLLING DATA TRANSFER FROM
20 REM BIOMATION MODEL 8100 TO AIM-65
30 REM
40 TABLE=3840
50 GOSUB 100:REM ARM THE BIOMATION
60 GOSUB 200:REM INITIALIZE REGISTERS IN ASSEMBLY ROUTINE
           AT $890
70 PRINT'TRIGGER THE BIOMATION TO LOAD A SIGNAL'
80 GOSUB 300:REM WAIT FOR SIGNAL TO BE STORED, THEN TAKE DATA
90 GOSUB 400:REM PRINT OUT OF DATA TABLE
99 END
100 POKE 40962,144:POKE 40960,0:POKE40960,1
110 RETURN
200 POKE 4,144:POKE 5,14
210 X=USR(N)
220 RETURN
300 PRINT'  ** WAIT  **'
310 IF PEEK(40973)<>16 THEN 310
320 POKE 4,176
330 X=USR(N)
340 IF PEEK(144)=3 THEN PRINT'  ** DATA TAKEN  **'
350 RETURN
400 FOR A = 0 TO 255 STEP 4
410 FOR B = 0 TO 3:PRINT PEEK(TA+A+B);
420 NEXT B:NEXT A
430 RETURN
500 END
```

After a fast analog signal is stored by the Biomation, 2K bytes of data are transferred into the microprocessor. A line diagram (figure 1) showing the voltage levels on each of the control lines is used to demonstrate how to transfer data over the eight data lines tied from the Biomation to Port A of the 6522.

CB1 is an input from the Biomation which indicates when the recording of a signal is finished. CB1 has been initiated as an input which will cause bit 4 of the IFR to be set when the recording has finished (negative transition). CB2 must then be held low to let the Biomation know that data is going to be taken. This is done by writing a %1100 0000 into the PCR, which puts CB2 into the manual output mode. CA1 has been programmed to set its flag (bit 1) in the IFR every time the Biomation drives that line high. The active transition from low to high causes an interrupt flag to be set because a 0 was put into bit 0 of the PCR.

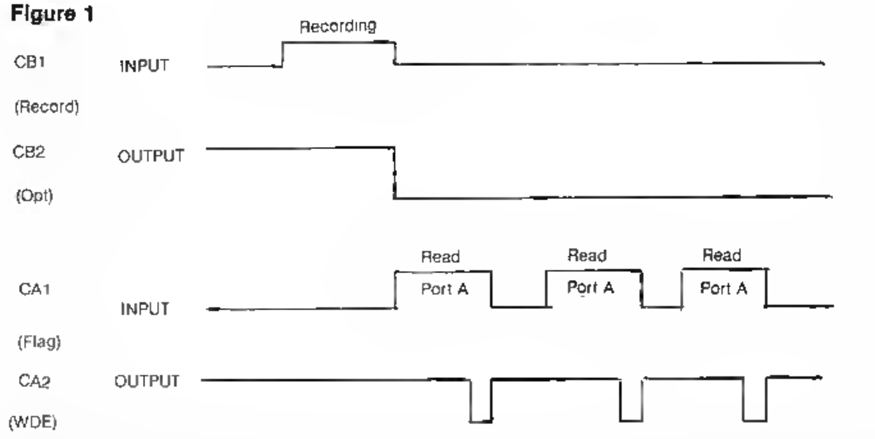
The program waits for the CA1 flag to go high in the IFR. When this happens, Port A is read by loading it into the accumulator. At the same time as Port A is read, the flag is automatically cleared and an output strobe appears on CA2. The output strobe indicates to the Biomation that a word was taken and it can make the next word available on the data pins. CA2 was initially set up by ORing the PCR with a %0000 1010.

The handshake data transfer technique is operated 2048 times until all of the data is transferred. After the last word is taken, CB2 is driven high to let

Table 1

BIOMATION J-10 Connector			AIM 65 J-1 Connector	
Name	Pin #		Pin #	Name
Y0	36	→	14	PA0
Y1	37	→	4	PA1
Y2	38	→	3	PA2
Y3	39	→	2	PA3
Y4	40	→	5	PA4
Y5	41	→	6	PA5
Y6	42	→	7	PA6
Y7	43	→	8	PA7
RMA	9	←	13	PB4
OFF	34	→	16	PB5
PPB	8	←	15	PB7
FLG	45	→	20	CA1
WDC	44	←	21	CA2
RECORD	50	←	18	CB1
OPT	7	→	19	CB2
GRND	1	–	1	GRND

Figure 1



the Biomation know that no more data will be read. The entire data transfer process normally requires about 0.2 seconds. This is an asynchronous data transfer because it is not under control of a clock.

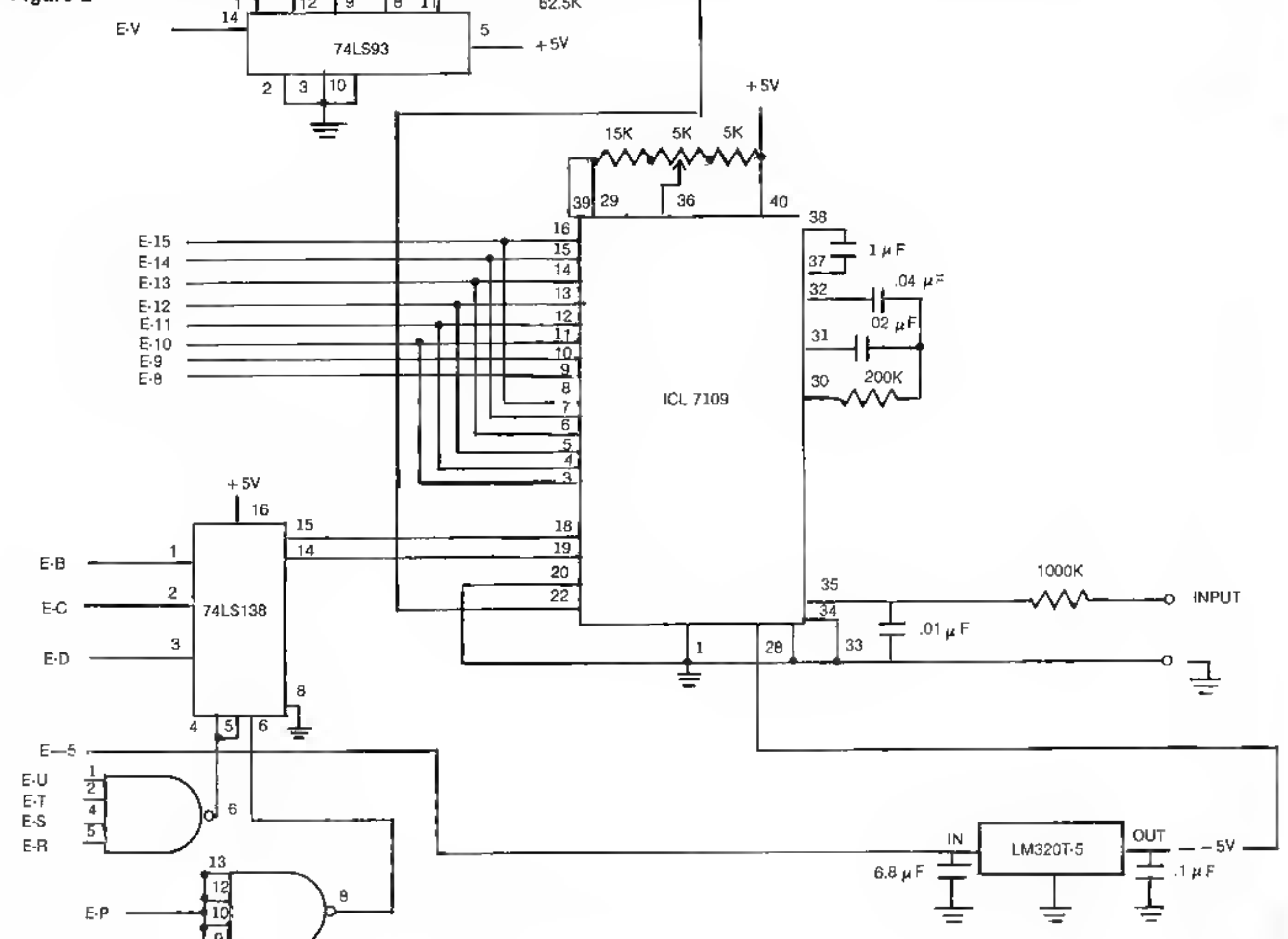
The assembly listing shows the initialization of the different registers. The first part of the program at $E90 is accessed from BASIC through a USR entry and then returns to BASIC by a RTS command. The BASIC program is used for operator interaction so that the user can start the waveform recorder and the data transfer at any time. The recording and the assembly program beginning at $EA3 can be started within seconds or minutes of each other and the data transfer will occur automatically. At the end of the transfer, program control will return to BASIC.

It should be apparent that this communication technique could easily be adapted for transferring information between two computers or between a computer and any instrument that has a resident ADC. As more and more of the analytical instruments become digital, binary word transfer mechanisms will become more prevalent. It is hoped that this model can be used to introduce the primary tools and information necessary to implement similar information transfers.

For those instrument applications where there are only analog signals to detect, the ADC scheme presented in figure 2, or in the other articles cited, should provide a method of communicating data to a microprocessor. The hardware outlined in the schematic requires some basic electronic knowledge and skills to get started. The hardware available on the SYM and AIM 65 can make any of these data transfers simpler if the programmer is aware of all of the available tools in these systems. There are many different ways to transfer data in and out of the MPU, with all of the equipment for doing this residing on the same board.

We'd like to thank Mr. Bill Stein for providing the hardware and software for the ICL7109.

Figure 2



MICRO

Taming the Wild Reset

Cut two traces, add two wires, and your old Apple reset key becomes a new Apple control-reset key!

Michael M. Seiy
Sytran One Inc.
1242 Home Avenue
Fort Wayne, IN 46807

This article describes a hardware modification of the Apple keyboard which will undoubtedly void any warranties. Neither the author nor the publisher will bear any liability for any damage you may do to your Apple while making these modifications. On the plus side, these modifications have been made on several Apples with no ill effects and in fact, are very similar to the changes which Apple has made to newer Apples.

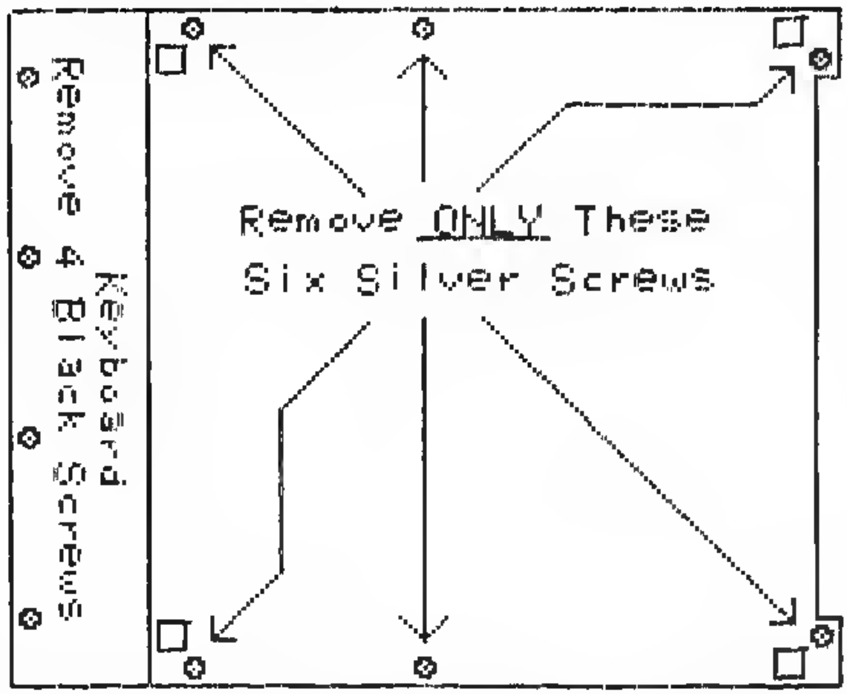
These instructions, which show you how to cut two traces and add two jumper wires, are written so that a novice who barely knows what a piece of wire is can tackle this project with confidence.

Once installed, your Apple will ignore all normal RESETS and acknowledge only a CONTROL-RESET with the deadly "BEEP *" we all dread.

You'll need the following tools to procede with these instructions:

- A #2 Phillips head screwdriver
- A sharp, small knife such as an X-acto
- A small soldering iron (25-40 watt)
- A short piece of electronic rosin core solder
- 12-15 inches of #22 or#24 insulated wire

Figure 1: Bottom view of Apple II.



The following are not necessary but may come in handy:

- A small (1/8") flat blade screwdriver
- Small needle-nose pliers

Step 1. Unplug the Apple and remove the cord from the case.

Step 2. Remove all cables from the back, such as the video and cassette cable.

Step 3. Remove the top cover of the Apple and remove all peripheral cards.

Step 4. Put the Apple upside down on a table top large enough to provide some elbow room.

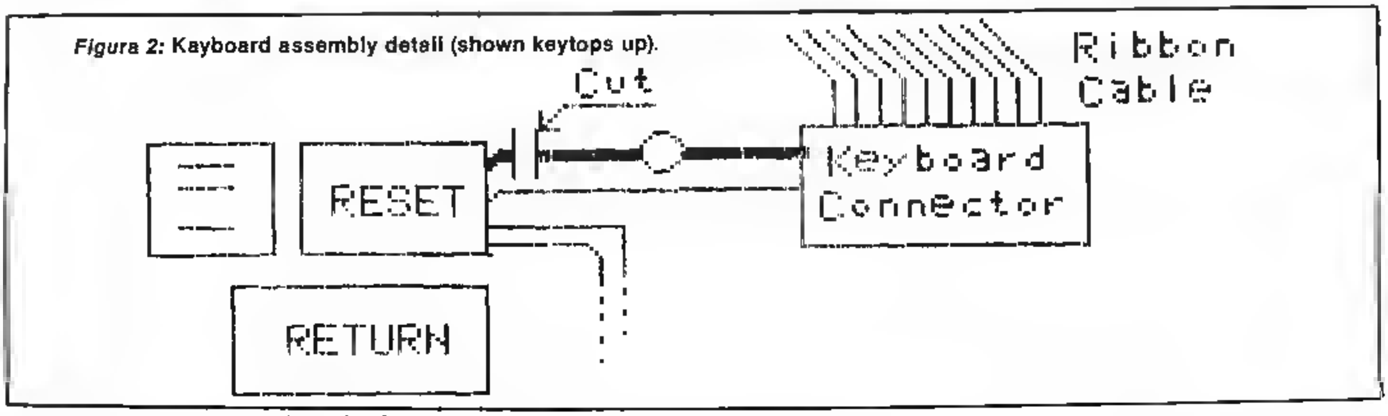
Step 5. Refer to figure 1 and remove the six flat-head, silver-colored Phillips head screws which hold the case to the bottom plate. They are all on the edge as shown. Do not remove any screws from the center.

Step 6. Remove the four Phillips screws at the front of the Apple.

Step 7. While holding the case and the bottom plate together, carefully turn the Apple rightside up, keyboard away from you. Look through the top of the case toward the front of the Apple and lift the case about two inches. You should spot the keyboard ribbon cable going to its connector on the main board. *Gently* rock the connector and ease it out of its socket. A small screwdriver may be used to pry the connector up. If you bend any pins, they should be straightened with needle nose pliers.

Step 8. Set the base plate aside for now and concentrate on the keyboard. Turn the case upside down, keyboard facing you, and remove the two screws on the left edge and the two screws on

Figura 2: Keyboard assembly detail (shown keytops up).



the right edge which hold the keyboard in the case. Remove the keyboard and set the case aside for now.

Step 9. With the keytops up (refer to figure 2), examine the area between the RESET key and the ribbon cable connector. You should see four printed circuit traces going under the RESET key on its right side. Using a sharp knife, cut through the larger upper (or rearward) trace as close to the reset key as possible. Be sure to cut all the way through the trace to break the electrical connection, but be careful not to damage any other traces.

Step 10. Turn the keyboard keytops down and refer to figure 3. Some Apples are equipped with an insulator taped at the lower left corner. This needs to be carefully peeled back. Locate the heavy circuit trace which runs along the bottom of the PC board. At the left edge it connects to a 40-pin I.C. Follow this trace up past the upper row of I.C. pins through contact #53, past three screws to the right side of contact #13. Cut this trace about ¼'' from the right contact of #13.

Step 11. Warm up a small soldering iron. Be sure to use rosin core solder for electronic applications and not the acid core solder used for plumbing.

Step 12. Cut a 9'' piece of wire and strip no more than 1/8'' of insulation from each end. Pre-tin the ends by heating them and applying small amounts of solder. Avoid melting the insulation as much as possible and if necessary, cut off some of the bare wire to keep its length at 1/8''.

Step 13. Cut and prepare a 3'' piece of wire the same as in step 12.

Step 14. Solder one end of the 9'' wire to the right contact of #13. This contact was connected to the circuit trace on the bottom of the board before the trace was cut in step 10.

Step 15. Locate the circuit trace at the bottom of the board again and this time follow it to the right. You should end up at the left contact of #28, which is on the back of the control key. Solder the other end of the 9'' wire to the empty right contact of #28.

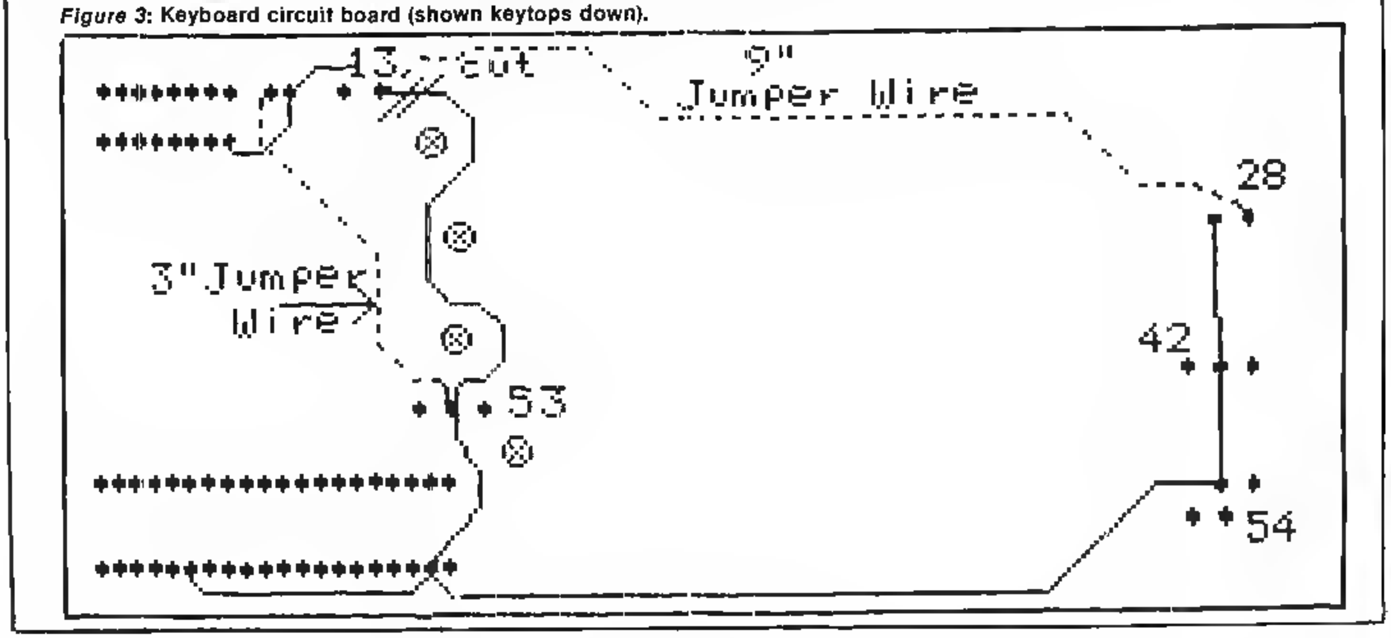
Step 16. Solder one end of the 3'' wire to the center contact of #53, thus connecting it to the circuit trace which runs along the bottom of the board.

Step 17. Between the contact pair #13 (used in steps 10 and 14) and the top row of contacts of the keyboard connector are two contacts. The right one has a trace connected to it. Solder the other end of the 3'' wire to the left, empty contact.

Step 18. Examine all of your solder connections for shorts to any nearby traces or terminals. Check for loose solder droplets and make sure the bare portion of the jumper wires does not touch any other circuit or terminal.

Step 19. Reposition the keyboard into its original place in the upper case. Be sure the ribbon cable is coming out from the top of the printed circuit card. Loosely fit each of the four screws which hold the keyboard into their respective holes and only then tighten them.

Figure 3: Keyboard circuit board (shown keytops down).



Step 20. Position the upper case over the main base and plug in the keyboard. Make sure that each pin goes into its place in the socket without bending and that the connector is not offset such that pins hang over the top or bottom of the socket.

Step 21. While holding the case and base together, turn the Apple upside down and refasten the case to the baseplate with the four black and six silver screws.

Step 22. Turn the Apple rightside up and reconnect the line cord. Do not, as yet, install any peripheral cards.

Step 23. Plug in the cord and turn on the Apple. You should hear the familiar RESET beep. Verify that pushing RESET does not produce a beep and that pushing CONTROL-RESET does. If you note any changes from this, turn off your Apple and recheck all jumper wires and cut traces. If the RESET key alone still does a RESET, one of the traces has not been cut all the way through or you cut the wrong one. If CONTROL-RESET does nothing, then the 9″ wire has a problem. If the entire keyboard does not respond, the 3″ wire has a problem.

Step 24. Turn the Apple off, re-install all of your peripheral cards and cables and you're back in business!

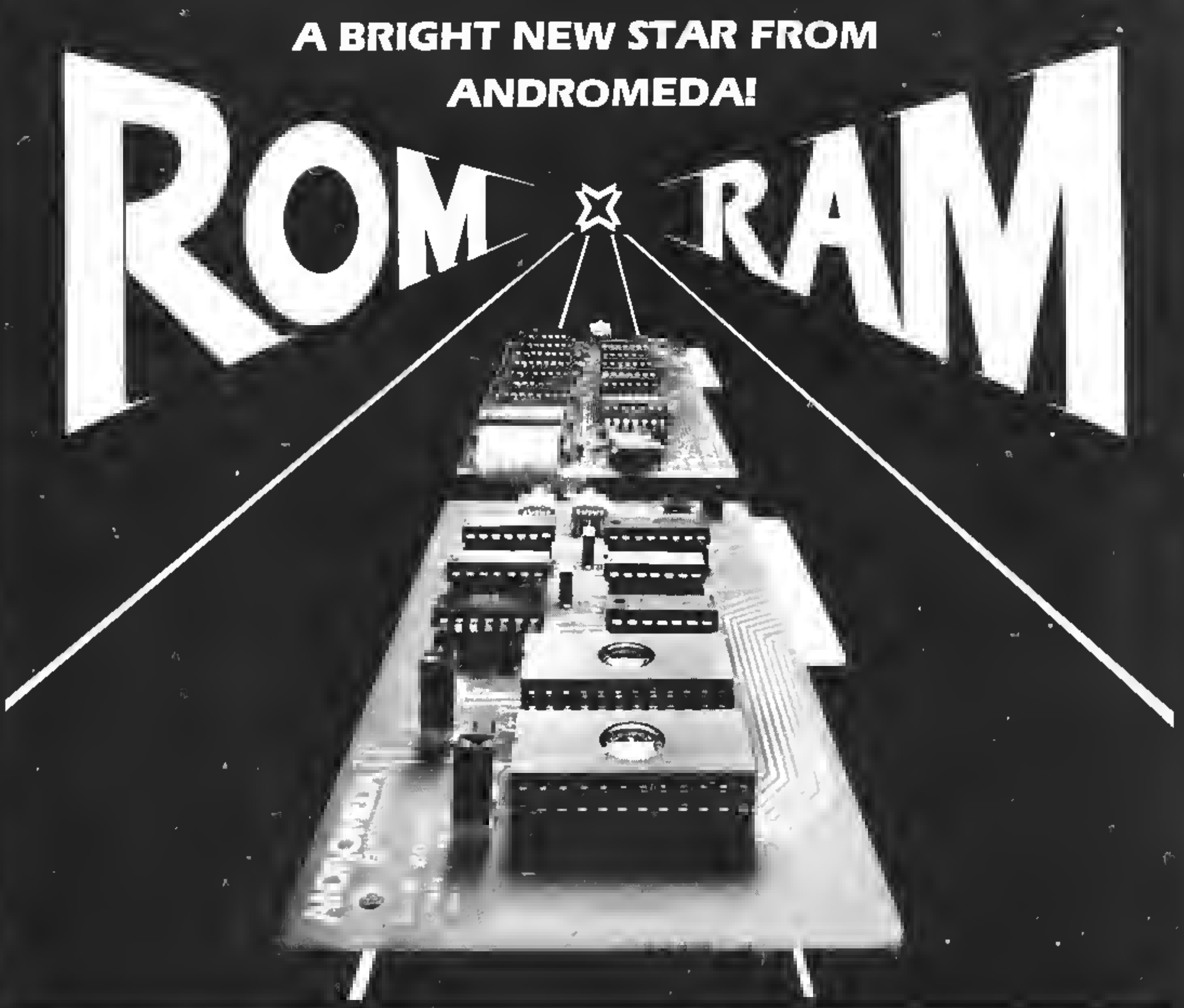
Theory of Operation

If you glance at the keyboard schematic on page 101 of the new *Apple Reference Manual*, you will see that the function of the RESET key is to connect the reset line from the Apple to ground. The control key also ties a line to ground. The two cut traces on the keyboard circuit card isolate the RESET key from circuit ground. The 9″ wire ties the RESET key to the floating side of the control key so that both keys must now be pressed to provide a path from the RESET line to ground.

Unfortunately, the cut traces also isolate the entire keyboard circuitry. The 3″ jumper wire restores ground to the rest of the keyboard.

Michael Seiy, a digital design engineer, has incorporated his own company which specializes in custom hardware and software for the Apple II. He owns a 48K Apple with two disk drives, an Applesoft card, a serial card, a parallel card, M.C. clock card, and an Epson printer.

MICRO



NEW ROM BOARD FOR THE APPLE II*
$125.00 WITH UTILITY ROM.

With Andromeda's new ROM Board, you can plug many useful utility programs into your Apple II. Because ROM memory never forgets, you can access these utilities instantly without having to load them from disk.

The ROM Board comes with the utility ROM, which gives you five powerful options to apply to your Applesoft* programs. With the Utility ROM, you can do automatic line numbering, control a program list with a page mode, restore a crashed Applesoft* program in memory, alphabetize a disk catalogue and create a disk without DOS, giving you an extra 8K on your disk. Any of Soft Control Systems' other ROMS can be used, such as the Dual DOS in ROM, and Your'ple ROM.

You can install 2K PROMS, 4K PROMS, or even 2K RAM chips in each of the two memory sockets. So you can even have the Read - Write capability of RAM to develop PROM Programs yourself, or just have an extra 2K RAM for your machine - Language programs. Two 2732 PROMS allow a total of 8K of memory on the Board.

Now with One Year Warranty.

Don't forget the Andromeda 16K RAM Expansion Board $195.00

ANDROMEDA

INCORPORATED

P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By

COMPUTER DATA SERVICES

P.O. Box 696
Amherst, NH. 03031
603 673-7375

* Apple II and Applesoft are trademarks

Apple Byte Table

This useful reference teble will simplify the tesk of decoding byte-velues in the Apple's memory. For ell numericel velues, hex or decimel, each possible meening is listed, renging from ASCII to Applesoft token. If you ever teckle e hex-dump, the Apple byte teble will prove inveluable.

Kim G. Woodward
6526 Delia Drive
Alexandria, Virginia 22310

If you look at a single byte in the Apple or any other 8-bit microcomputer, it will mean different things at different times. Data and instructions are represented in the same manner in the computer: one byte may be data, an address, a token, or a command. I have put together a simple table which will be of use no matter what the relationship of the byte is to your software. (Columns F, G, H, and I will be especially useful to the Apple owner.) The table is composed of 10 columns which represent:

A. The equivalent decimal value of the byte (assuming the byte is not signed).

B. The equivalent hex value of the byte.

C. The equivalent binary value of the byte (very useful for assembly language masking).

D. The value of the byte if it is looked at as the high byte of an address.

E. The corresponding ASCII character for the byte (if there is one).

F. The equivalent displayed screen character. (*I*-Inverse, *F*-Flashing, *N*-Normal.)

G. The equivalent key to be pressed to get the byte. (If there is one, note all keys > $7F, C after character means CTRL key held down.)

H. The corresponding Integer BASIC token for the byte. The Integer BASIC tokens can be found by keying:

```
> CALL -155          Go to monitor
 *CA:00 10           Set program start
 *4C:14 10           Set program end
 *1000:13            Set length byte
 *1001:0A 00         Set line number
 *1003:              16 bytes of your choice
 *1013:01            End of line token
 *                   Return via CTRL-C
>LIST
```

I. The corresponding Applesoft BASIC token for the byte. The Applesoft tokens can be found by keying:

```
CALL -155            Go to monitor
*67:01 08            Set program start
*AF:16 08            Set program end
*801:16 08           Pointer to next line
*803:0A 00           Set line number
*805:                16 bytes of your choice
*815:00              End of line token
*816:00 00 00        End of program pointer
*0G                  Back to BASIC
 LIST
```

J. The corresponding 6502 machine language opcode.

Let's note some of the subtleties in the table's usage. First of all, if a particular pattern for a mask operation is needed, then it is a simple matter of looking down the table until the correct binary (column 3) pattern is found. Then on the same line, read the decimal equivalent for a POKE command, or the hex equivalent for assembly language use. In like manner you can do the following:

A. Decimal to hexadecimal conversion — scan the table in column 4 to find the highest number not exceeding the decimal number. If the number is negative (such as addresses in Integer BASIC larger than 32767), add 65536 before the conversion. Write down the hex value and subtract the decimal number just found. Then find the decimal remainder in the table and write down the hex value for it. The first hex value is the high byte and the second is the low byte. For example, find the hex equivalent of −936 (clear).

−936 + 65536 = 64600 : the number to find. Find 64512 ($FC) : highest number less than 64600 -64512 = 88 : find difference. Find 88 ($58) : remainder. Value of −936 decimal is $FC58.

B. Hexadecimal to decimal conversion — separate the hex number into two bytes. Scan the table for the value of the high order byte in column 4. Then scan the table for the value of the low order byte in column 1, add the two numbers together and get the result. For negative addresses (> $7FFF) simply subtract 65536 from the number.

C. Relative addressing — the formula for relative addressing on the 6502 is: address of branch to address - address of branch inst. - 2. For example, to branch from location $345 to $313 you could find the decimal equivalent of $345 as per (A) above, 837, and of $313, 787. Thus 787 -837 - 2 is −52. Add 256 to −52 giving 204. Look up 204 in the table as $CC. $CC is then the relative address offset.

Columns F and G in the table can be found in the *Apple Reference Handbook* by Apple Computer, Inc. If these tables have been of some benefit, let me know; write to the address at the beginning of the article.

The Apple Byte Table

Dec	Hx	Binary	High	Asc	Sc	Ky	Int Bs	Aps Bs	6502
000	00	00000000	0	NUL	@I		HIMEM:	NUL	BRK
001	01	00000001	256	SOH	AI		EOS	SOH	ORAIX
002	02	00000010	512	STX	BI		_	STX	
003	03	00000011	768	ETX	CI		:	ETX	
004	04	00000100	1024	EOT	DI		LOAD	EOT	
005	05	00000101	1280	ENQ	EI		SAVE	ENQ	ORAZ
006	06	00000110	1536	ACK	FI		CON	ACK	ASLZ
007	07	00000111	1792	BEL	GI		RUN	BEL	
008	08	00001000	2048	BS	HI		RUN	BS	PHP
009	09	00001001	2304	HT	II		DEL	HT	ORAIM
010	0A	00001010	2560	LF	JI		,	LF	ASLA
011	0B	00001011	2816	VT	KI		NEW	VT	
012	0C	00001100	3072	FF	LI		CLR	FF	
013	0D	00001101	3328	CR	MI		AUTO	CR	ORA
014	0E	00001110	3584	SO	NI		,	SO	ASL
015	0F	00001111	3840	SI	OI		MAN	SI	
016	10	00010000	4096	DLE	PI		HIMEM:	DLE	BPL
017	11	00010001	4352	DC1	QI		LOMEM:	DC1	ORAIY
018	12	00010010	4608	DC2	RI		+	DC2	
019	13	00010011	4864	DC3	SI		−	DC3	
020	14	00010100	5120	DC4	TI		*	DC4	
021	15	00010101	5376	NAK	UI		/	NAK	ORAZX
022	16	00010110	5632	SYN	VI		=	SYN	ASLZX
023	17	00010111	5888	ETB	WI		#	ETB	
024	18	00011000	6144	CAN	XI		>=	CAN	CLC
025	19	00011001	6400	EM	YI		>	EM	ORAY
026	1A	00011010	6656	SUB	ZI		<=	SUB	
027	1B	00011011	6912	ESC	[I		<>	ESC	
028	1C	00011100	7168	FS	\I		<	FS	
029	1D	00011101	7424	GS	]I		AND	GS	ORAX
030	1E	00011110	7680	RS	^I		OR	RS	ASLX
031	1F	00011111	7936	US	_I		MOD	US	
032	20	00100000	8192	SPC	I		^	SPC	JSR
033	21	00100001	8448	!	!I		+	!	ANDIX
034	22	00100010	8704	"	"I		(	"	
035	23	00100011	8960	#	#I		,	#	
036	24	00100100	9216	$	$I		THEN	$	BITZ
037	25	00100101	9472	%	%I		THEN	%	ANDZ
038	26	00100110	9728	&	&I		,	&	ROLZ
039	27	00100111	9984	'	'1		,	'	
040	28	00101000	10240	(	(I		"	(	PLP
041	29	00101001	10496	)	)I		"	)	ANDIM
042	2A	00101010	10752	*	*I		(	*	ROLA
043	2B	00101011	11008	+	+I		!	+	
044	2C	00101100	11264	,	,I		!	,	BIT
045	2D	00101101	11520	−	−I		(	−	AND
046	2E	00101110	11776	.	.I		PEEK	.	ROL
047	2F	00101111	12032	/	/I		RND	/	
048	30	00110000	12288	0	0I		SGN	0	BMI
049	31	00110001	12544	1	1I		ABS	1	ANDIY
050	32	00110010	12800	2	2I		PDL	2	
051	33	00110011	13056	3	31		RNDX	3	
052	34	00110100	13312	4	4I		(	4	
053	35	00110101	13568	5	5I		+	5	ANDZX
054	36	00110110	13824	6	6I		−	6	ROLZX
055	37	00110111	14080	7	7I		NOT	7	
056	38	00111000	14336	8	8I		(	8	SEC
057	39	00111001	14592	9	9I		=	9	ANDY
058	3A	00111010	14848	:	:I		#	:	
059	3B	00111011	15104	;	;I		LEN(	;	
060	3C	00111100	15360	<	<I		ASC(	<	
061	3D	00111101	15616	=	=I		SCRN(	=	ANDX
062	3E	00111110	15872	>	>I		,	>	ROLX
063	3F	00111111	16128	?	?I		(	?	
064	40	01000000	16384	@	@F		$	@	RTI

(Continued)

Dec	Hx	Binary	High	Asc	Sc	Ky	Int Bs	Aps Bs	6502
065	41	01000001	16640	A	AF		*	A	EORIX
066	42	01000010	16896	B	BF		(	B	
067	43	01000011	17152	C	CF		,	C	
068	44	01000100	17408	D	DF		,	D	
069	45	01000101	17664	E	EF		;	E	EORZ
070	46	01000110	17920	F	FF		;	F	LSRZ
071	47	01000111	18176	G	GF		;	G	
072	48	01001000	18432	H	HF		,	H	PHA
073	49	01001001	18688	I	IF		,	I	EORIM
074	4A	01001010	18944	J	JF		,	J	LSRA
075	4B	01001011	19200	K	KF		TEXT	K	
076	4C	01001100	19456	L	LF		GR	L	JMP
077	4D	01001101	19712	M	MF		CALL	M	EOR
078	4E	01001110	19968	N	NF		DIM	N	LSR
079	4F	01001111	20224	O	OF		DIM	O	
080	50	01010000	20480	P	PF		TAB	P	BVC
081	51	01010001	20736	Q	QF		END	Q	EORIY
082	52	01010010	20992	R	RF		INPUT	R	
083	53	01010011	21248	S	SF		INPUT	S	
084	54	01010100	21504	T	TF		INPUT	T	
085	55	01010101	21760	U	UF		FOR	U	EORZX
086	56	01010110	22016	V	VF		=	V	LSRZX
087	57	01010111	22272	W	WF		TO	W	
088	58	01011000	22528	X	XF		STEP	X	CLI
089	59	01011001	22784	Y	YF		NEXT	Y	EORY
090	5A	01011010	23040	Z	ZF		,	Z	
091	5B	01011011	23296	[	[F		RETURN	[	
092	5C	01011100	23552	\	\F		GOSUB	\	
093	5D	01011101	23808	]	]F		REM	]	EORX
094	5E	01011110	24064	^	^F		LET	^	LSRX
095	5F	01011111	24320	_	_F		GOTO	_	
096	60	01100000	24576	'	F		IF		RTS
097	61	01100001	24832	a	!F		PRINT		ADCIX
098	62	01100010	25088	b	"F		PRINT		
099	63	01100011	25344	c	#F		PR1NT		
100	64	01100100	25600	d	$F		POKE		
101	65	01100101	25856	e	%F		,		ADCZ
102	66	01100110	26112	f	&F		COLOR=		RORZ
103	67	01100111	26368	g	'F		PLOT		
104	68	01101000	26624	h	(F		,		PLA
105	69	01101001	26880	i	)F		HLIN		ADCIM
106	6A	01101010	27136	j	*F		,		RORA
107	6B	01101011	27392	k	+F		AT		
108	6C	01101100	27648	l	,F		VLIN		JMPI
109	6D	01101101	27904	m	-F		,		ADC
110	6E	01101110	28160	n	.F		AT		ROR
111	6F	01101111	28416	o	/F		VTAB		
112	70	01110000	28672	p	0F		=		BVS
113	71	01110001	28928	q	1F		=		ADC1Y
114	72	01110010	29184	r	2F		)		
115	73	01110011	29440	s	3F		)		
116	74	01110100	29696	t	4F		LIST		
117	75	01110101	29952	u	5F		,		ADCZX
118	76	01110110	30208	v	6F		LIST		RORZX
119	77	01110111	30464	w	7F		POP		
120	78	01111000	30720	x	8F		NODSP		SEI
121	79	01111001	30976	y	9F		NODSP		ADCY
122	7A	01111010	31232	z	:F		NOTRACE		
123	7B	01111011	31488	{	;F		DSP		
124	7C	01111100	31744	¦	<FF		DSP		
125	7D	01111101	32000	}	=F		TRACE		ADCX
126	7E	01111110	32256	~	>F		PR#		RORX
127	7F	01111111	32512	RUB	?F		IN#		
128	80	10000000	32768		@N		NUL	END	
129	81	10000001	33024		AN	AC	SOH	FOR	STAIX
130	82	10000010	33280		BN	BC	STX	NEXT	
131	83	10000011	33536		CN	CC	ETX	DATA	
132	84	10000100	33792		DN	DC	EOT	INPUT	STYZ
133	85	10000101	34048		EN	EC	ENQ	DEL	STAZ
134	86	10000110	34304		FN	FC	ACK	DIM	STXZ
135	87	10000111	34560		GN	GC	BEL	READ	

Dec	Hx	Binary	High	Asc	Sc	Ky	Int Bs	Aps Bs	6502
136	88	10001000	34816		HN	HC	BS	GR	DEY
137	89	10001001	35072		IN	IC	HT	TEXT	
138	8A	10001010	35328		JN	JC	LF	PR#	TXA
139	8B	10001011	35584		KN	KC	VT	IN#	
140	8C	10001100	35840		LN	LC	FF	CALL	STY
141	8D	10001101	36096		MN	MC	CR	PLOT	STA
142	8E	10001110	36352		NN	NC	SO	HLIN	STX
143	8F	10001111	36608		ON	OC	SI	VLIN	
144	90	10010000	36864		PN	PC	DLE	HGR2	BCC
145	91	10010001	37120		QN	QC	DC1	HGR	STAIY
146	92	10010010	37376		RN	RC	DC2	HCOLOR=	
147	93	10010011	37632		SN	SC	DC3	HPLOT	
148	94	10010100	37888		TN	TC	DC4	DRAW	STYZX
149	95	10010101	38144		UN	UC	NAK	XDRAW	STAZX
150	96	10010110	38400		VN	VC	SYN	HTAB	STXZY
151	97	10010111	38656		WN	WC	ETB	HOME	
152	98	10011000	38912		XN	XC	CAN	ROT=	TYA
153	99	10011001	3916B		YN	YC	EM	SCALE=	STAY
154	9A	10011010	39424		ZN	ZC	SUB	SHLOAD	TXS
155	9B	10011011	39680		[N	ESC	ESC	TRACE	
156	9C	10011100	39936		\N		FSS	NOTRACE	
157	9D	10011101	40192		]N	MCU	GS	NORMAL	STAX
158	9E	10011110	40448		^N	^C	RS	INVERSE	
159	9F	10011111	40704		_N		US	FLASH	
160	A0	10100000	40960		N	SPC	SPC	COLOR=	LDYIM
161	A1	10100001	41216		!N	!	!	POP	LDAIX
162	A2	10100010	41472		"N	"	"	VTAB	LDXIM
163	A3	10100011	41728		#N	#	#	HIMEM:	
164	A4	10100100	41984		$N	$	$	LOMEM:	LDYZ
165	A5	10100101	42240		%N	%	%	ONERR	LDAZ
166	A6	10100110	42496		&N	&	&	RESUME	LDXZ
167	A7	10100111	42752		'N	'	'	RECALL	
168	AB	10101000	43008		(N	(	(	STORE	TAY
169	A9	10101001	43264		)N	)	)	SPEED=	LDAIM
170	AA	10101010	43520		*N	*	*	LET	TAX
171	AB	10101011	43776		+N	+	+	GOTO	
172	AC	10101100	44032		,N	,	,	RUN	LDY
173	AD	10101101	44288		−N	−	−	1F	LDA
174	AE	10101110	44544		.N	.	.	RESTORE	LDX
175	AF	10101111	44800		/N	/	/	&	
176	B0	10110000	45056		0N	0	0	GOSUB	BCS
177	B1	10110001	45312		1N	1	1	RETURN	LDAIY
178	B2	10110010	45568		2N	2	2	REM	
179	B3	10110011	45824		3N	3	3	STOP	
1B0	B4	101101000	46080		4N	4	4	ON	LDYZX
181	B5	10110101	46336		5N	5	5	WAIT	LDAZX
182	B6	10110110	46592		6N	6	6	LOAD	LDXZY

(Continued)

Dec	Hx	Binary	High	Asc	Sc	Ky	Int Bs	Aps Bs	6502
183	B7	10110111	46848		7N	7	7	SAVE	
184	B8	10111000	47104		8N	8	8	DEF	CLV
185	B9	10111001	47360		9N	9	9	POKE	LDAY
186	BA	10111010	47616		:N	:	:	PRINT	TSX
187	BB	10111011	47872		;N	;	;	CONT	
188	BC	10111100	48128		<N	<	<	LIST	LDYX
189	BD	10111101	48384		=N	=	=	CLEAR	LDAX
190	BE	10111110	48640		>N	>	>	GET	LDXY
191	BF	10111111	48896		?N	?	?	NEW	
192	C0	11000000	49152		@N	@	@	TAB(	CPYIM
193	C1	11000001	49408		AN	A	A	TO	CMPIX
194	C2	11000010	49664		BN	B	B	FN	
195	C3	11000011	49920		CN	C	C	SPC(	
196	C4	11000100	50176		DN	D	D	THEN	CPYZ
197	C5	11000101	50432		EN	E	E	AT	CMPZ
198	C6	11000110	50688		FN	F	F	NOT	DECZ
199	C7	11000111	50944		GN	G	G	STEP	
200	C8	11001000	51200		HN	H	H	+	INY
201	C9	11001001	51456		IN	I	I	−	CMPIM
202	CA	11001010	51712		JN	J	J	*	DEX
203	CB	11001011	51968		KN	K	K	/	
204	CC	11001100	52224		LN	L	L	^	CPY
205	CD	11001101	52480		MN	M	M	AND	CMP
206	CE	11001110	52736		NN	N	N	OR	DEC
207	CF	11001111	52992		ON	O	O	>	
208	D0	11010000	53248		PN	P	P	=	BNE
209	D1	11010001	53504		QN	Q	Q	<	CMPIY
210	D2	11010010	53760		RN	R	R	SGN	
211	D3	11010011	54016		SN	S	S	INT	
212	D4	11010100	54272		TN	T	T	ABS	
213	D5	11010101	54528		UN	U	U	USR	CMPZX
214	D6	11010110	54784		VN	V	V	FRE	DECZX
215	D7	11010111	55040		WN	W	W	SCRN(	
216	D8	11011000	55296		XN	X	X	PDL	CLD
217	D9	11011001	55552		YN	Y	Y	POS	CMPY
218	DA	11011010	55808		ZN	Z	Z	SQR	
219	DB	11011011	56064		[N		[	RND	
220	DC	11011100	56320		\N		\	LOG	
221	DD	11011101	56576		]N	MCU	]	EXP	CMPX
222	DE	11011110	56832		^N	^	^	COS	DECX
223	DF	11011111	57088		_N		_	SIN	
224	E0	11100000	57344		N			TAN	CPXIM
225	E1	11100001	57600		!N			ATN	SBCIX
226	E2	11100010	57856		"N			PEEK	
227	E3	11100011	58112		#N			LEN	
228	E4	11100100	58368		$N			STR$	CPXZ
229	E5	11100101	58624		%N			VAL	SBCZ
230	E6	11100110	58880		&N			ASC	INCZ
231	E7	11100111	59136		'N			CHR$	
232	E8	11101000	59392		(N			LEFT$	INX
233	E9	11101001	59648		)N			RIGHT$	SBCIM
234	EA	11101010	59904		*N			MID$	NOP
235	EB	11101011	60160		+N				
236	EC	11101100	60416		,N			SYNTAX	CPX
237	ED	11101101	60672		−N			RWO GSB	SBC
238	EE	11101110	60928		.N			OUT DTA	INC
239	EF	11101111	61184		/N			ILL QNT	
240	F0	11110000	61440		0N			OVERFLW	BEQ
241	F1	11110001	61696		1N			OUT MEM	SBCIY
242	F2	11110010	61952		2N			UNF STM	
243	F3	11110011	62208		3N			BD SUBS	
244	F4	11110100	62464		4N			RDM ARY	
245	F5	11110101	62720		5N			DIV ZER	SBCZX
246	F6	11110110	62976		6N			ILL DIR	INCZX
247	F7	11110111	63232		7N			TYP MIS	
248	F8	11111000	63488		8N			STR LNG	SED
249	F9	11111001	63744		9N			FRM CPX	SBCY
250	FA	11111010	64000		:N			CANTCNT	
251	FB	11111011	64256		;N			UNDFNC	
252	FC	11111100	64512		<N			ERROR	
253	FD	11111101	64768		=N			(	SBCX
254	FE	11111110	65024		>N			(	INCX
255	FF	11111111	65280		?N			(	

Apple Bits, Part 2

Part 1 of last month's article, "Apple Bits," (40:75) presented a machine language program for plotting low resolution graphics patterns from encoded data. This part will present an Integer BASIC program for constructing the patterns used by the machine language driver. Next month's part will give detailed instructions on how to create low resolution animations using these tools.

Richard C. Vile
3467 Yellowstone Drive
Ann Arbor, MI 48105

Figure 1: **Building the pattern.**

The Pattern Maker Program

The program allows you to create patterns and store them in tables for subsequent use by animation programs. It begins by asking a couple of questions:

HEIGHT AND WIDTH OF PATTERNS?
TABLE ADDRESS IN DECIMAL?

The patterns created may be *up to* 8 rows high by 8 columns wide, but may be smaller than that as well. For example, one set of patterns that I use consists of 7 rows by 5 columns. They form a "giant" character set that may be used to create billboard messages on the Apple screen. The table of patterns is stored in Apple RAM and manipulated by PEEKs and POKEs. Thus, it is necessary to tell the program where in memory the table is located. This is the reason for the second question. I typically store tables at 3072 ($C00). The tables must be saved on tape or disk for eventual use by animation programs.

The program will display a rectangular border enclosing an area equal in size to the patterns specified, as shown in figure 1. Inside the pattern border, a blinking cursor will be seen. The user may move this cursor about, inside the border, and either add or delete parts of a pattern in the process.

The pattern maker will respond to any of the following commands:

PATTERN
VERIFY
MODIFY
RECORD
SAME
HELP
QUIT,BYE,STOP,EXIT

The commands are typed in full, or abbreviated to the first letter. If you forget what the commands are, simply type "HELP" or "H" and the menu of commands will be listed for you. (Note: You will probably lose any pattern in progress if you do that.)

The commands have the following effects:

PATTERN: The area inside the border is erased, the cursor appears inside, and the user may begin creating a new pattern.

MODIFY: Recalls a given pattern from the table, so the user may modify it.

SAME: Returns to the same pattern as the one most recently created or modified (allows the user to recover from accidently striking "ENTER" while creating a pattern.)

VERIFY: Displays the numeric codes for the pattern under construction or modification. Mainly included for debugging the pattern maker program itself.

HELP: Displays the menu of commands.

QUIT,BYE,STOP,EXIT: Cause the termination of the program. Note that the screen is cleared and returned to TEXT mode.

The program operates in mixed low resolution graphics mode and uses the bottom four lines of the screen for entering commands and prompts. The program will prompt the user by typing:

COMMAND?

and then waiting for a response. If any of the above commands are entered, the program will take the corresponding action, otherwise it simply reprompts the user. The "P", "M", and "S" commands will cause the cursor to be transferred inside the rectangle on the graphics portion of the screen. While there, the user may enter "cursor control keys" or "pattern control keys" to shift the cursor around the pattern and create or erase parts of the pattern.

The cursor control keys and their results are listed in table 1 and the pattern control keys and their results are listed in table 2.

Table 1

KEY	EFFECT
→	Move the cursor one column to the right. If the cursor is already at the far right of the rectangle, then "wrap" around to the far left of the pattern, but one row further down. If at the extreme bottom right of the pattern, then "wrap" around to the extreme top left of the pattern.
R	Same as →.
←	Move the cursor one column to the left. At the extreme positions "wrap" around in a fashion analogous to that described above for the → or R keys.
L	Same as ←.
U	Move the cursor up one row. (Wrap around also)
D	Move the cursor down one row. (Wrap also)
ENTER	Return the cursor to the command area of the screen.
ESC	Same as for "ENTER".

Table 2

KEY	EFFECT
+	Add a solid blob to the pattern in the position indicated by the current location of the cursor.
–	Erase the part of the pattern (if any) located at the current position of the cursor.

Pattern Creation Utility

```
 0 REM  PATTERN MAKER PROGRAM FOR APPLE LORES GRAPHICS
 1 DIM PTTERN(7),BITS(7),A$(25)
 2 FOR I=0 TO 7:PTTERN(I)=BITS(I)=0: NEXT I
 4 KBD=-16384:CLR=-16368:WAIT=3000
 5 GOSUB I0000
10 INPUT "COMMAND? ",A$
I1 IF A$="P" OR A$="PATTERN" THEN GOSUB 50
I2 IF A$="V" OR A$="VERIFY" THEN GOSUB 1000
13 IF A$="M" OR A$="MODIFY" THEN GOSUB I500
14 IF A$="R" OR A$="RECORD" THEN GOSUB 2000
15 IF A$="S" OR A$="SAME" THEN GOSUB 52
t6 IF A$="H" OR A$="HELP" THEN GOSUB 2500
17 IF A$="Q" OR A$="QUIT" OR A$="BYE" OR A$="STOP" OR A$="EXIT"
   THEN GOTO 3025
45 GOTO 10
50 FOR I=0 TO 7:PTTERN(I)=0: NEXT I: GR
51 COLOR=1: HLIN I4,14+WIDTH+I AT 14: HLIN I4,I4+WIDTH+t AT 14+
   HEIGHT+1: VLIN 14,14+HEIGHT+1 AT 14: VLIN 14,14+HEIGHT+1 AT
   I4+WIDTH+t
52 SAVCOLR= SCRN(15+COL,15+ROW):KEY= PEEK (KBD): IF KEY>=I28 THEN
   58
54 COLOR=15: PLOT I5+COL,15+ROW: FOR I=0 TO 10: NEXT I: COLOR=0
   : PLOT t5+COL,I5+ROW: FOR I=0 TO 10: NEXT I: IF SAVCOLR#15 THEN
   52
56 COLOR=I5: PLOT 15+COL,15+ROW: COLOR=0: GOTO 52
58 IF KEY=I4I OR KEY=155 THEN RETURN : POKE CLR,0: COLOR=15
60 IF KEY# ASC("R") AND KEY#t49 THEN 70:COL=COL+1: IF COL<WIDTH
   THEN 52:ROW=ROW+I:COL=0: IF ROW<HEIGHT THEN 52:ROW=0: GOTO 52
70 IF KEY# ASC("L") AND KEY#136 THEN 80:COL=COL-1: IF COL>=0 THEN
   52:COL=WIDTH-I:ROW=ROW-1: IF ROW>=0 THEN 52:ROW=HEIGHT-1:COL=
   WIDTH-1: GOTO 52
80 IF KEY# ASC("U") THEN 90:ROW=ROW-1: IF ROW>=0 THEN 52:ROW=
   HEIGHT-1:COL=COL-I: IF COL>=0 THEN 52:COL=WIDTH-I: GOTO 52
90 IF KEY# ASC("D") THEN I00:ROW=ROW+I: IF ROW<HEIGHT THEN 52:ROW=
   0:COL=COL+I: IF COL<WIDTH THEN 52:COL=0: GOTO 52
t00 IF KEY# ASC("+") THEN 110:VALUE=I: GOSUB 500: GOTO 52
tI0 IF KEY# ASC("-") THEN t20:VALUE=0: GOSUB 500: GOTO 52
120 VTAB 23: PRINT "INVALID KEY": FOR K=1 TO 25: NEXT K: VTAB 23
   : TAB t: PRINT "              ": GOTO 52
```

When RECORDing or MODIFYing patterns, the program will request a KEY to associate with the pattern. The user should respond to this request by simply striking the desired key (do not hit ENTER, unless that is the desired key). Control keys (except for Control-c) are included. The association that is made "internally" by this is as follows: The program converts the ASCII value of the key struck to a table offset. This offset is then used when storing or retrieving the corresponding pattern from memory. The same idea will be used by animation programs in order to point the machine language driver at the correct positions in memory for a given pattern.

The pattern maker program does not LOAD and SAVE the pattern tables itself. This is the responsibility of the user. For example, suppose you have created a table which starts at location \$C00 and extends as far as \$FFF. After exiting the pattern maker program and returning to the Integer BASIC command level, you would give the following command:

```
> BSAVE PATTERN TABLE XYZ,
A$C00,L$7FF
```

assuming that you have a disk-based system. To save the same table on tape, you would enter the monitor and (after setting up your recorder, etc.) type:

```
*C00.FFFW
```

and wait for the monitor to write it all out to the cassette.

Below is the listing of the program in Integer BASIC. *Note:* If you store your tables in low memory, be sure to protect them from the BASIC program itself. For example, when I use the area from \$C00 (decimal 3072) to \$FFF, I first issue the command:

```
LOMEM: 4096
```

Final Note

The pattern maker program uses the machine language driver program (in order to support the Modify command). Thus, the following complete sequence of commands would be used to run the pattern maker to add or modify patterns previously saved in file BPATS:

```
> BLOAD BPATS
> BLOAD APPLE-BITS
> LOMEM: 4096
> RUN PATTERN MAKER
```

If no previous file of patterns, such as BPATS, is being used, then the first command in the sequence may be omitted.

```
 500 TEMP=PTTERN(COL)
 510 FOR B=0 TO 7:BITS(B)=TEMP MOD 2:TEMP=TEMP/2: NEXT B
 515 BITS(ROW)=VALUE
 517 TEMP=BITS(7)
 520 FOR B=6 TO 0 STEP -1
 530 TEMP=2*TEMP+BITS(B)
 540 NEXT B
 550 PTTERN(COL)=TEMP
 551 IF VALUE=0 THEN COLOR=0
 555 PLOT 15+COL,15+ROW
 557 COLOR=15
 560 RETURN
1000 FOR I=0 TO 7: PRINT PTTERN(I);" ";: NEXT I
1010 RETURN
1500 INPUT "WHICH KEY?"
1505 KEY= PEEK (KBD): IF KEY<128 THEN 1505
1510 POKE CLR,0:OFFSET=(KEY-128)*WIDTH
1512 POKE 2048,WIDTH: POKE 2049,HEIGHT
1515 POKE 60,(ADDR+OFFSET) MOD 256
1520 POKE 61,(ADDR+OFFSET)/256
1522 GR
1525 POKE 36,15: POKE 37,15
1530 COLOR=15: CALL 2058
1532 POKE 36,0: POKE 37,23
1535 COLOR=1: HLIN 14,14+WIDTH+1 AT 14: HLIN 14,14+WIDTH+1 AT 14+
     HEIGHT+1
1540 VLIN 14,14+HEIGHT+1 AT 14: VLIN 14,14+HEIGHT+1 AT 14+WIDTH+1

1545 FOR I=0 TO WIDTH-1
1550 PTTERN(I)= PEEK (ADDR+OFFSET+I)
1555 NEXT I
1560 GOTO 52
2000 PRINT "WHICH KEY?"
2001 KEY= PEEK (KBD): IF KEY<128 THEN 2001
2002 POKE CLR,0:KEY=KEY-128:OFFSET=KEY*WIDTH
2005 FOR I=0 TO WIDTH-1
2010 POKE ADDR+OFFSET+I,PTTERN(I)
2020 NEXT I
2030 RETURN
2500 REM  HELP SUBROUTINE
2501 REM
2510 TEXT : CALL -936
2515 VTAB 2: TAB 2: PRINT "COMMAND";: TAB 12: PRINT "EFFECT"
2520 TAB 2: PRINT "=======";: TAB 12: PRINT "======="
2525 VTAB 5: TAB 2: PRINT "PATTERN";: TAB 12: PRINT "STARTS A NEW
     PATT ERN"
2526 PRINT
2527 TAB 2: PRINT "MODIFY";: TAB 12: PRINT "CALLS UP AN OLD PATTERN
     FO R"
2529 TAB 12: PRINT "MODIFICATIONS."
2530 PRINT
2531 TAB 2: PRINT "RECORD";: TAB 12: PRINT "SAVES CURRENT PATTERN
     IN T HE"
2533 TAB 12: PRINT "PATTERN TABLE. IT WILL BE"
2535 TAB 12: PRINT "ASSOCIATED WITH A KEY."
2536 PRINT
2537 TAB 2: PRINT "SAME";: TAB 12: PRINT "RETURNS TO PATTERN AREA"

2539 TAB 12: PRINT "WITHOUT DESTROYING THE"
2541 TAB 12: PRINT "CURRENT PATTERN."
2542 PRINT
2543 TAB 2: PRINT "HELP";: TAB 12: PRINT "DISPLAYS THIS MESSAGE."

2585 PRINT : TAB 2: PRINT "  TO QUIT, TYPE ANY OF THE FOLLOWING:"

2587 TAB 2: PRINT "  'QUIT','Q','STOP','BYE', OR 'EXIT'"
2590 GOSUB WAIT
2599 RETURN
3000 REM  WAIT SUBROUTINE
3001 REM
3005 POKE CLR,0
3010 KEY= PEEK (KBD): IF KEY<128 THEN 3010
3015 POKE CLR,0
3020 IF KEY# ASC("Q") THEN RETURN
3025 TEXT : CALL -936: END
10000 TEXT : CALL -936
10005 KBD=-16384:CLR=-16368
10010 INPUT "HEIGHT OF PATTERNS ",HEIGHT
10015 INPUT "WIDTH OF PATTERNS ", WIDTH
10020 INPUT "TABLE ADDRESS IN DECIMAL",ADDR
10030 RETURN
```

MICRO

WHY THE MICROSOFT RAMCARD™ MAKES OUR SOFTCARD™ AN EVEN BETTER IDEA.

Memory — you never seem to have quite enough of it.

But if you're one of the thousands of Apple owners using the SoftCard, there's an economical new way to expand your memory dramatically.

16K ON A PLUG-IN CARD.

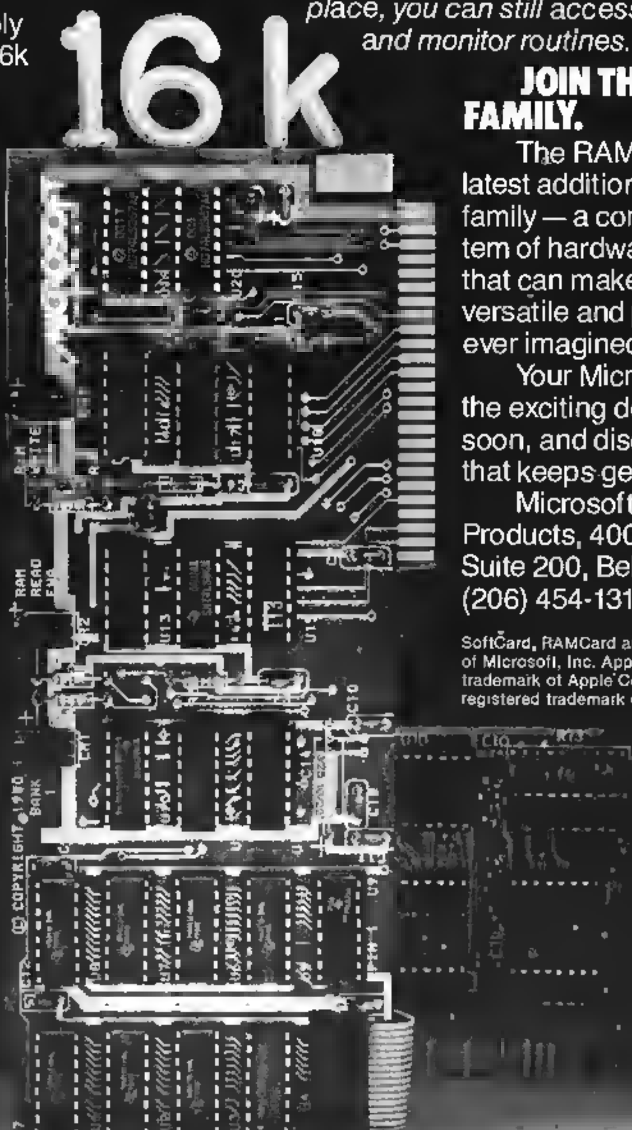
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

Together with the SoftCard, the RAMCard gives you a 56k CP/M® system that's big enough to take on all kinds of chores that would never fit before (until now, the only way to get this much memory was to have an Apple Language Card installed).

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

With the RAMCard and SoftCard, you can tackle large-scale business and scientific computing with our COBOL and FORTRAN languages. Or greatly increase the capability of CP/M applications like the Peachtree Software accounting systems. VisiCalc™ and other Apple software packages can take advantage of RAMCard too.

And RAMCard gives you the extra capacity to develop advanced programs of your own, using the SoftCard and CP/M. *Even with the RAMCard in place, you can still access your ROM BASIC and monitor routines.*



JOIN THE SOFTCARD FAMILY.

The RAMCard is just the latest addition to the SoftCard family — a comprehensive system of hardware and software that can make your Apple more versatile and powerful than you ever imagined.

Your Microsoft dealer has all the exciting details. Visit him soon, and discover a great idea that keeps getting better.

Microsoft Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

SoftCard, RAMCard and Microsoft are trademarks of Microsoft, Inc. Apple II is a registered trademark of Apple Computer Inc. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research Inc. VisiCalc is a registered trademark of Personal Software, Inc. Microsoft Consumer Products is a division of Microsoft, Inc.

MICROSOFT CONSUMER PRODUCTS

MICROSOFT™

MICRO PET Vet

By Loren Wright

It seems that many of those accessories promised for the VIC are really going to appear in October. The VIC 1515 graphic printer is now available, and at a relatively low price — $395. It is based on the Seikosha printer. The Seikosha printer, mentioned in its Axiom GP-80M implementation in our printer overview in August (MICRO 39:33), is a dot matrix with an interesting design. A single hammer strikes rapidly against splines on a platen, which rotate freely beneath the paper. This printer prints the entire PET/VIC character set, and any programmable characters you might come up with. It is also pin programmable so you can produce high resolution, dot graphic printouts. The design compromises made, perhaps accounting for the low price, are slow printing speed (30 characters per second) and its requirement for a special, narrower paper (8" pin to pin). Fortunately, the paper shouldn't be too expensive since it is plain, not heat sensitive or aluminum coated.

A number of expansion cartridges and games will be released in September and October. Some of the games have working names like "VIC Slot," "Jupiter Lander," and VIC Avenger." On the more serious side, there will be a programmable character generator program, a machine language monitor, a Toolkit-like "Programmer's Aid," and a package called "Super Expander." The "Super Expander" expands VIC BASIC to include things like convenient color, circle-drawing, and music-playing commands. Prices start at $30 for some of the games.

Also in October, the 3K and 8K RAM expansion cartridges will be available. The 16K expansion will require the expansion module, which probably won't be available until early next year. The *Programmer's Reference Manual*, which contains memory maps and a more thorough documentation of BASIC, is now available for $14.95. The Commodore light pen, mentioned in David Malmberg's article (page 54, this issue) will not be available in the near future, so people wanting to try Mr. Malmberg's programs will have to use the Systems Formulate or Atari light pens.

My previous VIC announcements apparently gave some people the false impression that MICRO is involved in selling the VIC. We received a few letters asking for information but, unfortunately, no checks!

Commodore Hotlines

Apple tried having a toll-free hot line number, but when they were deluged with calls in the first few months, the company was forced to cancel the service and change to a system where the numbers are available only to dealers. I called Commodore to find out if they, too, were having second thoughts, but I was encouraged to publicize the two numbers.

For technical information only:

1-800-523-5622

For all other information:

1-800-523-5614

MICRO

Interfacing Two 12-Bit A/D Converters to an AIM

Use 12-bit A/D converters for extra precision. A BASIC program on AIM 65 may be used to call a machine code routine to run the converters for logging applications.

G. Roger Heal and
J. Derek Openshaw
Department of Chemistry and
Applied Chemistry
University of Salford
Salford M5 4WT, England

An important application of microprocessors is in data logging systems where analog signals representing temperature, pressure, flow, weight, etc., are converted into digital codes and stored or displayed. The AIM 65 microcomputer is a particularly useful device for these systems because the timers in the on-board VIA (versatile interface adaptor) can be used to control the logging interval, and the built-in printer is ideal for recording the data.

The subject of A/D converters has been introduced by Marvin De Jong, MICRO 15:40. His device was of the 8-bit type. This gives a precision of readings of 1 in 256 or approximately 0.4%. For many purposes this may be sufficiently accurate, but for scientific use, higher precision is usually required. Twelve bits gives a precision of 1 in 4096 or approximately 0.025%. The 7570 J device, described by De Jong, uses the successive approximations method of conversion and is fast. In general, the price of A/D converters increases with number of bits and with the speed of operation. For our purpose, speed of conversion was unimportant, but precision was. If calculations with the logged data are to be carried out in BASIC, between the logging operations, this executes relatively slowly and the value of a fast A/D converter may be lost.

For these reasons we chose to use an ICL 7109 manufactured by INTERSIL. This is a 12-bit device with tri-state buffers for direct connection to microprocessor data lines. It uses the dual slope integrating technique and so the conversion is not particularly fast. It is quoted as working at up to 30 conversions per second, but we ran it much slower than that. It has the advantage of only costing about L10 in the United Kingdom. The manufacturer's data sheet on the device helpfully gives several methods of interfacing to microprocessors. Since the AIM 65 has eight data lines, the 12 bits from the converter have to be transferred in two bytes, first as an 8-bit byte followed by 4 bits in a second operation. The first 8 bits transferred are the least significant bits (right-hand part of the number) or lo-byte, and the remaining four bits are the most significant bits or hi-byte. With these four bits are a polarity signal, POL, which is high if the voltage being converted is negative, and an over-range signal, OR, which is high if the voltage applied to is too high for the device to convert.

In our application we required two analog signals representing temperature and weight to be recorded. We could have used a multiplexing system feeding to one A/D converter. However, since the devices described are so cheap, we decided to use separate converters for each channel. Since our analog signals were changing very slowly, we avoided the need for sample and hold amplifiers before the converters. Thus, since we had to deal with two converters and two blocks of data from each, the interface wiring and controlling program had to be carefully designed to sequence the operations in the correct order. The interface controlling subroutine is written in 6502 assembly language, entered via the AIM editor and assembled using the on-board ROM assembler. This machine code subroutine is called from a BASIC main program with the BASIC USR function. The assembled machine code program is, in practice, stored on a cassette tape, as is the main BASIC program.

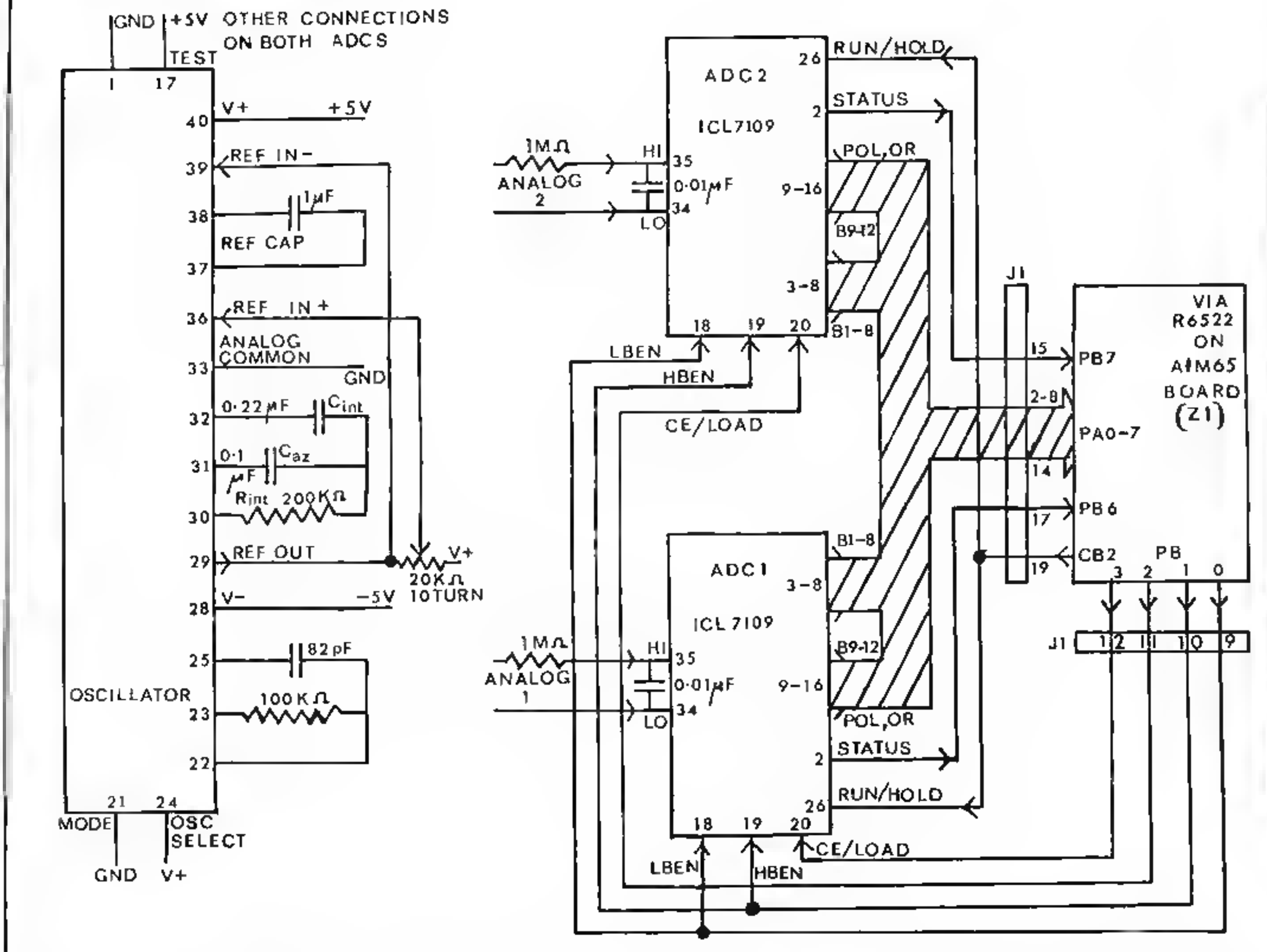
Wiring the Interface

The ICL 7109 is in a 40-pin dual-in-line package and is wired using a 40-pin socket and supplied with the required ±5 volt power leads. The interface between the two A/D converters and the AIM 65 is shown in figure 1. The connections are via J1, the user interface plug on the AIM 65 board, which is linked directly to the user VIA, R6522 (Z1 on the AIM board). Some connections are the same on both 7109s and are shown once on the left-hand part of the diagram. These connections include the power supplies (GND, +5V, −5V), integrating capacitor C_{INT}, auto-zero capacitor C_{AZ}, reference capacitor C_{ref}, and oscillator components C_o and R_o. A built-in reference voltage is taken out from pin 29 to a 10-turn potentiometer to provide the reference voltage to pin 39 for the second, de-integration stage of conversion. An external reference voltage could be provided, but this reference from pin 29 was found to be stable enough for most purposes. Several modes of operation are possible with these converters. With pin 21 taken to ground, the "direct" mode is selected. This causes the output data bits to be only latched into the data lines under the control of $\overline{CE/LOAD}$ (chip select) and $\overline{HBEN}$ and $\overline{LBEN}$ (byte enable signals).

The internal oscillator may also be controlled in two ways. With pin 24 taken to the +5V line, an RC configuration is used. (With pin 24 taken to ground, crystal control is selected.) The RC values shown give an oscillator frequency of about 70 KHz, which allows about 10 conversions per second if run at full speed continuously. The exact conversion time depends upon the magnitude of the signal and the operation of the RUN/HOLD line on pin 26.

If RUN/HOLD is taken high for at least seven periods of the oscillator, the first stage of conversion (integration) starts and RUN/HOLD may then be taken back to a low level. If RUN/

***Figure 1:* Control and Data Lines**



HOLD is held continuously high, the converter will cycle continuously, i.e. starting a new conversion as soon as the previous one has finished. If, however, RUN/HOLD is taken low after the conversion has started, the conversion is finished and the device will halt in an auto-zero mode, with the digital data ready for access. In the layout shown, the RUN/HOLD lines for both A/D converters, pin 26, are linked to the control output CB2 of the VIA. The machine code program is set up to take CB2 to high (binary 1) for a short period and back to low (binary 0) again.

The microcomputer must be able to determine if the A/D converters have finished their separate conversions. To do this their STATUS lines (pin 2) are linked to the input port lines PB6 and PB7 for converter 1 and converter 2 respectively. The STATUS signal goes high when conversion starts (integration) and goes low again when the conversion has finished and the data has been stored in the output latches. The program is arranged to check PB6 and PB7 until they are both low, before continuing on to read the data from the latches.

When the data is available from both converters, the four steps of data transfer start. The lo-byte (B1-B8), hi-byte (B9-B12), POL and OR lines from both converters are linked together into data lines PA0-PA7 on the VIA. The wiring pattern is shown in table 1. The data lines PB0-PB3 are used to apply chip enable and byte enable signals to the converters. These lines are normally all held high and no data is transferred. When both converters are ready the first converter is selected by taking PB3 low. Therefore its CE/LOAD is taken low. Simultaneously lo-byte is selected by taking PB0 low. The first data byte is then read off from lines PA0-PA7 and stored in RAM. If PA0-PA7 are configured as input lines, then the microcomputer reads then as if they were ordinary memory at address $A001. Thus, LDA $A001 puts the data into the accumulator, where it may be stored or operated upon.

Next PB0 is restored to high and PB1 is taken low to transfer hi-byte, POL and OR. Note that lines PA4, PA5 are not connected at this moment, so their signals are redundant. POL and OR are deliberately connected to PA7, PA6 respectively, because there are convenient program instructions for checking the state of these lines using the N and V flags in the PSR (program status register). The second converter is now selected by PB2 and the same two steps for PB0 and PB1 are used for the transfer of its data.

The analog signals are connected to pins 34 and 35, with pin 34 normally at ground potential, and should be provided with the 1M Ω resistor and 0.01μ F capacitor to filter out AC noise. The analog signal amplitude may be up to twice the reference voltage applied to

Table 1: **Data bus connections.**

Data Bit or Function	ICL 7109 Pin Number	Connector J1 Pin Number	VIA Port Connection
1	16	14	PA0
2	15	4	PA1
3	14	3	PA2
4	13	2	PA3
5	12	5	PA4
6	11	6	PA5
7	10	7	PA6
8	9	8	PA7
9	8	14	PA0
10	7	4	PA1
11	6	3	PA2
12	5	2	PA3
OR (overrange)	4	7	PA6
POL (polarity)	3	8	PA7

pin 39 to produce a full scale reading. In the circuit shown, adjustment of the 10 turn potentiometer gives full scale readings in the range 1-2 volts with reasonable accuracy.

Assembly Program

The machine code program is to be stored at the top end of the available RAM, leaving the rest for the BASIC program. The starting address is set to $0F00 (highest address $0FE6). The listing in assembly language is given in listing 1. The first part, entered at E1, is an initializing routine and only needs to be entered once. A second entry point, labelled E2, is the normal entry point to start the converters and read their data. Timer 1, in the VIA, is used to determine the period between conversions and makes use of the internal AIM interrupt line IRQ. This causes a jump to a routine addressed by an interrupt vector. The entry point for this interrupt routine is at VEC. From this point the processor runs through four instructions to RTI and then returns to the normal program. The address for this entry point VEC has to be set in $A400 (lo-byte) and $A401 (hi-byte). The intializing routine carries this out automatically. The instructions ·BYT<VEC and ·BYT>VEC load the low byte and high byte parts of the address of VEC into the locations VEC1 and VEC2 respectively. The first steps of the initializing routine then load VEC1 into $A400 and VEC2 into $A401. This method also has the advantage that, if the program instructions are changed during development work, the new entry point address of VEC is automatically re-calculated.

Next a $00 is put into $A003 (DDRA, Data Direction Register A) which configures all the PA lines of the VIA as inputs. The pattern of bits %0011 1111 represented by $3F is then put into $A002 (DDRB, Data Direction Register B) to configure PB6 and PB7 as inputs, but PB0 to PB5 as outputs (PB4 and PB5 are actually not used). The converters are started by RUN/HOLD going high then low after a short delay, as explained above. If the oscillator frequency is about 70 KHz, a time of 100 μ s is required to span 7 cycles. This may be conveniently achieved by the use of the delay routine in the AIM monitor, located at $EC0F.

This routine uses an on-board timer in the R6522 monitor VIA (Z32) and the timing period has to be set in $A418 (lo-byte) and $A417 (hi-byte) in multiples of the AIM clock time (1 μs approximately). In the initalizing routine a time of $0060 96 decimal is set. It appears that this period, 96 μs, is slightly too short, but in practice the instructions before and after the delay is called, take up additional time themselves, so the real delay is well over 100 μs. It should be noted that this timer on the R6522 monitor VIA is the one used to time the width of bits on the serial interface to peripherals from the AIM. Therefore a teletype could not be used with this arrangement.

The next instruction sets the bit pattern %0100 0000 ($40) in $A00B (ACR, Auxiliary Control Register) which sets the timer T1 on the R6522 user VIA to free running mode and allows continuous interrupts, without output pulses from PB7. The basic timing period for T1 now has to be set by writing into $A004 (T1L-L, lo-byte) and $A005 (T1L-H, hi-byte). If the maximum value of $FFFF were used this would correspond to 65, 535 μs (decimal) or 65.535 ms. It was decided to shorten this to exactly 50 ms for convenience. The hexadecimal pattern corresponding to this, $C350, was loaded and the period of the timer was measured using an accurate frequency meter attached to the IRQ interrupt line on the AIM. It was found that the period was slightly too long, presumably because the crystal on our particular AIM gave a period slightly higher than the nominal 1 μs. In the interests of accuracy, trial and error were used to lower the timer period to exactly 50 mS. A final setting of $C33F (49983 decimal) was arrived at and $C3 and $3F are set in $A005 and $A004 respectively. Actually, writing to $A004 puts the low byte into a latch and writing to $A005 puts the high byte into a second latch and transfers both latch values into the timer itself and starts the timer. At the same time the T1 interrupt flag is cleared. The 50 mS period is too short on its own, so multiples of this are required, produced by a counter in the program. A suitable example would be to count 40 interrupts, which would produce a 2-second interval.

When the machine code routine is entered from the BASIC instruction USR (N), the value of N is left in the FPA (floating point accumulator). Calling the routine at $BEFE converts this number into an integer in a single byte at $AD. This value may be put into COUNT and COUNTB. COUNTB is used for the actual counting, but may be reset from the value held in COUNT.

Listing 1

```
                  ;* ASSEMBLY LANGUAGE VERSION
                  ;*     OF USR SUBROUTINE
                  ;*
                         ORG $0F00
                  ;
                  ;INITIAL ENTRY
                  ;
0F00 ADE50F       E1     LDA VEC1          ;IRQ ENTRY
0F03 8D00A4              STA $A400         ;ADDR LO
0F06 ADE60F              LDA VEC2          ;ADDR HI
0F09 8D01A4              STA $A401
0F0C A900                LDA #$00
0F0E 8D03A0              STA $A003         ;PA AS INPUT
0F11 A93F                LDA #$3F          ;PB6,PB7-IN
0F13 8D02A0              STA $A002         ;REST PB OUT
0F16 A960                LDA #$60          ;SET DELAY
0F18 8D18A4              STA $A418         ;TIME FOR
0F1B A900                LDA #$00          ;SUB $EC0F
0F1D 8D17A4              STA $A417
0F20 A940                LDA #$40          ;T1 FREE RUN
0F22 8D0BA0              STA $A00B         ;CONT. INT., PB7 DISABLE
0F25 A93F                LDA #$3F          ;WRITE T1L-L
0F27 8D04A0              STA $A004
0F2A A9C3                LDA #$C3          ;WRITE T1L-H,T1C-H
0F2C 8D05A0              STA $A005         ;T1L-L-T1C-L,CLR T1 INT.FLAG
0F2F 20FEBE              JSR $BEFE         ;FLOAT TO
0F32 A5AD                LDA $AD           ;INTEGER
0F34 8DE30F              STA COUNT         ;AND STORE
0F37 8DE40F              STA COUNTB
0F3A 58                  CLI               ;CLEAR INTERRUPT
0F3B A9C0                LDA #$C0          ;ENABLE T1
0F3D 8D0EA0              STA $A00E         ;INTERRUPTS
0F40 4CD1C0              JMP $C0D1         ;RETN BASIC
0F43              ;
0F43              ;NORMAL ENTRY
0F43              ;
0F43 20FEBE       E2     JSR $BEFE         ;NORMAL ENTRY
0F46 A5AD                LDA $AD           ;SEE IF 1/2
0F48 D04A                BNE L4            ;READ
0F4A ADE40F       RETN   LDA COUNTB        ;IS IT DOWN
0F4D F003                BEQ LOG           ;TO ZERO
0F4F 4C4A0F              JMP RETN          ;LOOP BACK
0F52 ADE30F       LOG    LDA COUNT         ;RUN ADCS
0F55 8DE40F              STA COUNTB        ;RESET
0F58 A9E0                LDA #$E0          ;TAKE CB2 HI
0F5A 8D0CA0              STA $A00C
0F5D 200FEC              JSR $EC0F         ;DELAY SUB
0F60 A9C0                LDA #$C0          ;TAKE CB2 LO
0F62 8D0CA0              STA $A00C
0F65 2C00A0       L1     BIT $A000         ;TEST IF ADCS FIN
0F68 5003                BVC L2            ;IF PB6=0
0F6A 4C650F              JMP L1
0F6D 2C00A0       L2     BIT $A000
0F70 1003                BPL L3            ;IF PB7=0
0F72 4C6D0F              JMP L2
0F75 A906         L3     LDA #$06          ;SET ADC1,
0F77 8D00A0              STA $A000         ;LO BYTE
0F7A AC01A0              LDY $A001         ;READ PA
0F7D 8CE10F              STY NUM           ;MAYBE NEG
0F80 A905                LDA #$05          ;SET ADC1,
0F82 8D00A0              STA $A000         ;HI BYTE
0F85 2C01A0              BIT $A001         ;TEST PB6,PB7
0F88 7029                BVS ERR           ;IF PA6 SET
0F8A 3035                BMI NEG           ;IF PA7 SET
0F8C AD01A0              LDA $A001         ;READ PA
0F8F 290F                AND #$0F          ;MASK 4 BIT
0F91 4CD1C0              JMP $C0D1         ; RETN BASIC
0F94 A90A         L4     LDA #$0A          ;2ND READ:SET ADC2,
0F96 8D00A0              STA $A000         ;LO BYTE
0F99 AC01A0              LDY $A001         ;READ PA
0F9C 8CE10F              STY NUM
0F9F A909                LDA #$09          ;SET ADC2,
0FA1 8D00A0              STA $A000         ;HI BYTE
0FA4 2C01A0              BIT $A001         ;TEST PA6,PA7
0FA7 700A                BVS ERR           ;IF PA6 SET
0FA9 3016                BMI NEG           ;IF PA7 SET
0FAB AD01A0              LDA $A001         ;READ PA
0FAE 290F                AND #$0F          ;MASK 4 BIT
0FB0 4CD1C0              JMP $C0D1         ;RETN BASIC
0FB3 20F0E9       ERR    JSR $E9F0         ;OUT CR/LF
```

(Continued)

The initializing routine now clears all interrupts on the AIM and enables T1 interrupts by writing the pattern %1100 0000 ($C0) to $A00E. Finally a return to BASIC is made via the routine $C0D1 (within the BASIC interpreter).

Each time the timer T1 (which counts downwards) reaches zero, an interrupt occurs. This causes the program control to finish the present instruction and transfer to the program instructions at VEC. Here, the accumulator is saved on the stack then $A004 (T1C-L) is read. This is a dummy read to clear the T1 interrupt. Note that when T1 reaches zero the count in the latches is reset into the counter and counting continues. The value of COUNTB is also reduced by 1 and the accumulator pulled back from the stack. Finally a return is made from the interrupt to the original program.

The main BASIC program enters the second routine at E2. The parameter of USR, left in the FPA again, is converted to integer using $BEFE and the value in $AD is tested. If it is zero, it is assumed

```
0FB6 A956                LDA #$56        ;ASCII V
0FB8 20BCE9              JSR $E9BC       ;OUT CHAR
0FBB 20F0E9              JSR $E9F0       ;OUT CR/LF
0FBE 4CD1C0              JMP $C0D1       ;RETN BASIC
0FC1 AD01A0      NEG     LDA $A001       ;NEG NUM READ-READ PA
0FC4 290F                AND #$0F        ;MASK 4 BIT
0FC6 8DE20F              STA NUM2        ;STORE
0FC9 38                  SEC             ;SET CARRY
0FCA A900                LDA #$00
0FCC EDE10F              SBC NUM         ;TWOS COMPL.
0FCF A8                  TAY             ;PUT IN Y
0FD0 A900                LDA #$00        ;TWOS COMPL.
0FD2 EDE20F              SBC NUM2        ;WITH BORROW
0FD5 4CD1C0              JMP $C0D1       ;RETN BASIC
0FD8             ;
0FD8             ;IRQ ENTRY
0FD8             ;
0FD8 48          VEC     PHA             ;STORE A
0FD9 AD04A0              LDA $A004       ;READ T1C-L
0FDC CEE40F              DEC COUNTB      ;CLEARS T1 INTERRUPT
0FDF 68                  PLA             ;RESTORE A
0FE0 40                  RTI             ;RETN FROM IRQ
0FE1             ;
0FE1             ;VARIABLE AREA
0FE1             ;
0FE1 00          NUM     BYT $00
0FE2 00          NUM2    BYT $00
0FE3 00          COUNT   BYT $00
0FE4 00          COUNTB  BYT $00
0FE5 D8          VEC1    BYT VEC         ;LO BYTE OF VEC ADDRESS
0FE6 0F          VEC2    HBY VEC         ;HI BYTE OF VEC ADDRESS
                         END
```

that the converters have not been started and there is no data available. The program then tests COUNTB for zero. If it is not, the small loop via RETN is cycled continuously until the interrupt routine has reduced COUNTB to zero, then a jump to LOG is made. At this point COUNTB is reset to the value in COUNT so a new timer period is started immediately. Writing the pattern %1110 0000 ($E0) to $A00C (peripheral control register) takes CB2 high, hence RUN/HOLD high. The delay via $EC0F is now called. Then the pattern %1100 0000 ($C0) is written to $A00C to take CB2 low again. Note, only 1 bit has been changed, but the others have to be held with the values shown to give CB2 this particular configuration, leaving CA1, CA2, CB1 unused.

A small loop is now entered at L1, looking at $A000 (port B data register). When PB6 (linked to STATUS of converter 1) has gone low, this is detected because the instruction BIT causes the bit 6 to be loaded into the overflow flag V of the PSR (processor status register). When bit 6 is zero the instruction BVC causes a branch to L2. A second loop here tests PB7, because bit 7 is loaded into the negative flag N of the PSR and a zero is registered as a positive number, so that BPL causes a branch to L3. Both converters have now stopped and the data may be read.

The pattern %0000 0110 ($06) is now written to $A000 (port B data register) to cause the lo-byte data from the first converter to appear on the PA lines. The data is read from $A001 (port A data register) and put into the Y register.

Next the pattern %0000 0101 produces the most significant bits on the PA lines. The BIT instruction first tests for over-range or a negative number. These conditions cause a jump to routines to either print a letter V to show overflow, or to take the two's complement of the number (hi and lo-byte). This latter routine produces a normal negative number on return to BASIC. If there is no special condition, then the hi-byte data is read, the lower four bits masked off and left in the accumulator. With the lo-byte data still in Y, a jump to $C0D1 converts to a standard floating point BASIC number and also returns to BASIC interpretation. A second call of USR causes entry at E2 again. If the parameter is not zero (e.g. 1) the converters are not started, but a branch to L4 then reads the data from the second converter by writing suitable bit patterns to $A000. The entry point E1 is set at $0F00 and, after assembly, the entry E2 is found to be at $0F43.

***Listing 2:* Example of a BASIC program.**

```
10 N=20
20 POKE4,0:POKE5,15
30 M=USR(N)
40 POKE4,67
50 X=USR(0)/400
60 Y=USR(1)/400
70 PRINT X,Y
80 GOTO 50
90 END
```

BASIC Program

The POKE instructions set the entry points to the machine code program and require addresses in decimal. Thus 0 (decimal) is $00 and 15 (decimal) is $0F. Also 67 (decimal) is $43.

A simple example of a BASIC program is shown in listing 2. Line 20 sets the first entry point for the USR routine called in line 30 to initialize the interface. The assignment of a value to X here is a dummy, but N sets the timing period in multiples of 50 ms. Line 40 resets the entry point to E2, then line 50 starts the converters and transfers the first result to X. Next, line 60 transfers the value from the second converter to Y. The two values are then printed or displayed and the program cycles back to read two more values. There will, of course, be an accurately timed delay in line 50 until 20X 50 ms (1 second) have elapsed.

The BASIC program used is very much under the control of the user. Other variations could be to print only every ten readings, say, to slow the timing, or to use polynomial smoothing on a set of pairs of points to remove random noise, only printing the midpoint of the set. If the reference voltage is adjusted so that the maximum voltage to be read produces a reading of 4000, then division of 400 gives values printed ranging up to 10.000 for convenience.

In our application we preceded the A/D converters by operational amplifiers to amplify 10 mV (max) signals to the 2-volt level. The reference voltage was then set at 1 volt. The two signals being logged were from a thermocouple which produced about 10 mV at 1000°C and an output from the control unit of a thermal balance, which produced 10 mV corresponding to 10 mg. Thus the 10.000 printed by the program represented 10 mV for channel and 10 mg weight for the other. Later variations were to convert the thermocouple EMF to temperature in degrees and, since the temperature was continuously raised, to print rate of change of temperature and weight.

It should be noted that entering BASIC by instruction 5 (initial) and replying with a RETURN to the question MEMORY SIZE? will cause all available store to be filled with the character A. Thus the machine code program should be loaded first, then BASIC should be entered and a memory size to prevent the overwriting of the machine code should be specified. A value of 2000 (decimal) is sufficient for a small BASIC program, but the maximum (up to $0EFF) 3839 (decimal) could be entered for larger ones.

Roger Heal is a lecturer in physical chemistry at Salford and teaches applications of microprocessors in chemistry to undergraduate courses. He has had experience over many years in the use of mainframe and minicomputers. Derek Openshaw is the scientific officer in charge of the Chemistry Department's electronic workshop and has worked on linking microprocessors to many chemical instruments.

MICRO

Dithertizer II™...

...the eye of your apple.®

Though it is very simple to use, the Dithertizer II represents the ultimate in video digitizing using the Apple II computer. The Dithertizer is an interface card which converts video input into digitized images. Because the Dithertizer II is a frame grabber, DMA type digitizer, it offers extreme high speed in the conversion process (it grabs an entire frame in 1/60th of a second). The camera supplied with the package is the Sanyo model VC1610X. Cabling is supplied for this camera so as to have the Dithertizer II system up and running in minutes. The video camera used for input must have external sync to allow for the frame grabber technology employed for digitizing. If a camera other than the model recommended is used, wiring adaptations by the user may be required. Software is supplied with the board to allow you to display up to 64 pseudo grey levels on your Apple's screen. The number of grey levels may be changed with one keystroke. The intensity and contrast of the image are controllable via game paddles. Also supplied is software for image contouring for those interested in movement detection or graphic design applications.

The Dithertizer II package is available ready to run with camera, interface card and the software described above for only: **$650.00**

Dithertizer II interface card and software (without camera): **$300.00**

Computer Station
11610 Page Service Dr.
St. Louis, MO 63141
(314) 432-7019

Apple II is a registered trademark of Apple Computer, Inc.
Ditherthizer II is a trademark of Computer Station, Inc.

THE ultimate in SPEED and language POWER for the APPLE II :*

THE INTEGER BASIC COMPILER

- SPEED IMPROVEMENT BY A FACTOR OF 10 TO 20 OVER APPLE'S BASIC INTERPRETER - 15 TO 30 OVER APPLESOFT!
- NO LANGUAGE CARD NECESSARY !
- OPTIMIZE CODE FOR YOUR SPEED/SPACE REQUIREMENTS.
- OBJECT CODE AND RUN-TIME SYSTEM ARE COMPLETELY RELOCATABLE - USE MEMORY THE WAY YOU WANT TO !
- MANY POWERFUL BASIC LANGUAGE EXTENSIONS:
  - FULL STRING LENGTH OF 32767 - NO 255 LIMIT !
  - CHR$, GET AND KEY FUNCTIONS.
  - DIRECT HI-RES GRAPHICS SUPPORT.
  - HOME, INVERT, NORMAL, FLASH, AND MORE !
- MANY APPLICATIONS - EXISTING INTEGER PROGRAMS CAN BE EASILY CONVERTED TO RUN ON ANY APPLE II !
- COMPILER REQUIRES: APPLE II (OR II PLUS WITH INTEGER OR LANGUAGE CARD), 48 K AND DOS 3.3.
- SUPPLIED ON 2 DISKS WITH COMPLETE DOCUMENTATION.

PRICE: $ 149.50

Dealer inquiries invited.

(Calif. add 6.5% sales tax. Foreign add $5.00 air mail)

GALFO SYSTEMS 6252 CAMINO VERDE
SAN JOSE, CA 95119

* Apple, Applesoft - Trademarks of Apple Computer Co.

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203

SOFTWARE FOR THE APPLE II*

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your **BASIC** programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Use advanced logic constructs such as **IF...ELSE...**, **CASE**, **SELECT**, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of **PASCAL**.
$35 Disk, Applesoft (48K, ROM or Language Card)

DSA–DS is a dis-assembler for 6502 code. Now you can easily dis-assemble any machine language program for the Apple and use the dis-assembled code directly as input to your assembler. Dis-assembles instructions and data. Produces code compatible with the S-C Assembler (version 4.0), Apple's Toolkit assembler and others.
$25 Disk, Applesoft (32K, ROM or Language Card)

FORM-DS is a complete system for the definition of input and output forms. **FORM-DS** supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
$25 Disk, Applesoft (32K, ROM or Language Card)

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's **CLEAR** gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
$25 Disk, Applesoft

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, **SPEED-DS** includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
$15 Disk, Applesoft (32K, ROM or Language Card).

(Add $4.00 for Foreign Mail)

*Apple II is a registered trademark of the Apple Computer Co.

Apple II

sensational software

creative computing software

Story Time and Don't Fall

Disk CS-4702 $24.95 Requires 16K Apple II or Apple II Plus

Story Time.
What's your name? Please name someone you like. What are you afraid of? Name a food you don't like. Using the answers to these questions, the computer makes up a delightful story presented with sound and full-color graphics. Over 100,000 variations are possible which incorporate different locales (desert, mountain, seashore, school yard), weather, time of day, and surprise events (find a creepy cave, haunted house, pink flying saucer and much more).

Don't Fall
Several new features make this adaptation of hangman better than ever. You can choose words from nine categories (sports, countries, etc.). Every letter you guess that is not in the words brings you one step closer to falling in the water. In the second part of each round you give the computer a word which it must guess, otherwise it falls in. Three skill levels.

Available 5/81

Know Yourself

Cassette CS-4301 $11.95 5 Programs Requires 16K Integer Basic

Alcohol Using body weight, time drinking, and type and number of drinks the program predicts your likely behavior.

Computer Literacy 5 questions per round on history, technology and practices in the computer field.

Sex Role Are you androgynous? Examine your nature, behavior and attitudes in light of society's changing concept of sex roles.

Life Expectancy The computer analyzes your responses to 20 questions and estimates your life expectancy. Find out the influence of where you live, smoking or exercise with this program.

Psychotherapy Feelings, phobias and mental health are analyzed in this program. The computer makes no judgements but helps you decide if you might benefit from a psychotherapist.

3 Adventures

Disk CS-4513 $39.95
Requires 48K Apple II or Apple II Plus

Adventureland (by Scott Adams) You'll encounter wild animals, dwarfs and many other puzzles and perils as you wander through an enchanted world, trying to rescue the 13 lost treasures. Can you rescue the Blue Ox from the quicksand? Or find your way out of the maze of pits? Happy Adventuring!

Pirate Adventure (by Scott Adams)— Yo Ho Ho and a bottle of rum... You'll meet up with the pirate and his daffy bird along with many strange sights as you attempt to go from your London flat to Treasure Island. Can you recover Long John Silver's lost treasures? Happy sailing matey.

Mission Impossible Adventure (by Scott Adams)— Good Morning. Your mission is to ... and so it starts. Will you be able to complete your mission in time? Or is the world's first automated nuclear reactor doomed? This one's well named, its hard, there is no magic but plenty of suspense.
Good Luck.

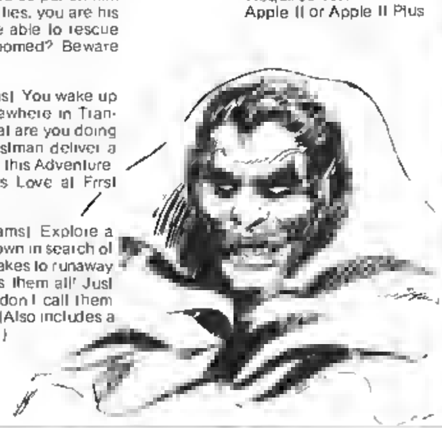
Voodoo Castle The Count and Ghost Town

Disk CS-4514 $39.95
Requires 48K
Apple II or Apple II Plus

Voodoo Castle (by Scott Adams) Count Cristo has had a fiendish curse put on him by his enemies. There he lies, you are his only hope... will you be able to rescue him—or is he forever doomed? Beware the Voodoo man.

The Count (by Scott Adams) You wake up in a large brass bed somewhere in Transylvania. Who are you, what are you doing here, and why did the postman deliver a bottle of blood? You'll love this Adventure. In fact, you might say it's Love at First Byte.

Ghost Town (by Scott Adams) Explore a deserted western mining town in search of 13 treasures. From rattlesnakes to runaway horses, this Adventure has them all! Just remember, pardner, they don't call them Ghost Towns for nothin'! (Also includes a new bonus scoring system.)



Adventures on Cassette

Five adventures are available singly on cassette. Each requires a 16K Apple II or Apple II Plus and costs $14.95.

CS-4011 Adventureland
CS-4012 Pirate Adventure
CS-4013 Mission Impossible
CS-4014 Voodoo Castle
CS-4015 The Count

Story Time

Cassette CS-4207, $11.95
Requires 16K Apple II or Apple II Plus

Available 5/81

Apple is the registered trademark of Apple Computer Inc.

Order Today

To order any of these software packages, send payment plus $2.00 postage and handling per order to Creative Computing, Morris Plains, NJ 07950, Attn: Colleen.
Visa, MasterCard and American Express orders may be called in toll-free.

Order today at no risk. If you are not completely satisfied, your money will be promptly and courteously refunded.

Creative Computing Software
Morris Plains, NJ 07950
Toll-free **800-631-8112**
In NJ 201-540-0445

creative computing software

Solar System Simulation

Part 2

Dave Partyka
1707 N. Nantuckett Drive
Lorain, Ohio 44053

This program will print information, or plot positions about the first six planets of the Solar System. In the printing mode, information such as distance from the earth and sun, and other data about the earth and planet relation is printed. In the plot mode, the planets' positions against the zodiac, as seen from the earth, are plotted, using Hi-Res graphics and scaling factors. Each mode uses planet choices, starting date, length of time, and time intervals to give any desired simulation.

The last program I wrote, ''Solar System Simulation With or Without an Apple II'' in the August '80 issue of MICRO (27:33) dealt with the orbits of the first six planets of the solar system. Their positions were plotted in reference to the sun, as seen from a point located quite a distance above the solar system. This was OK if you wanted to see the relation of each planet at specific times, or for various lengths of time. However, watching that simulation didn't show what was happening as seen from the earth. After phone calls from many users of that program who asked if I had other programs dealing with astronomy, I decided to write another one.

This program also deals with the first six planets, but instead of being heliocentric (sun centered) it's geocentric (earth centered). It uses calculations that I had in the first program, along with other ones, to give a display of the planets as seen from the earth. The planets are displayed against a star background and their motions through the zodiac are very good representations of the actual positions of the planets. Using this program, you can watch as a planet makes its retrograde loop through a constellation, see how close two or more planets come to each other, or watch how close a planet comes to a bright star.

I won't go into detail about the calculations used in this program. Some are the same as in the other program, and I have explained them there. What I will do is explain what this program does, the questions that it asks, describe the star table and how to expand it, and explain the splitting of this program to make another.

The program is set up in two parts. One part prints values on the screen for each planet and the sun, and the other plots the positions of the planets against a star background. If you choose to print, at the top of the screen is the starting date and the number of days that the display is for. The program then prints the following data for each planet:

- D-S; the distance, in million miles that the planet is from the sun.
- A-S; the angle in degrees that the planet is located around the sun.
- D-E; the distance in million miles that the planet is from the earth.
- R.A.; the right ascension in hours and minutes that the planet appears from the earth.
- DEC.; the declination in degrees and minutes that the planet appears from the earth.

You can display the values for all the planets, or for specific ones. You can display a single day, or a range of days with any number of days between the displays. The program will pause after each display, and then wait for you to press RETURN to continue with the display, or with a set of questions for a new display.

If you choose to plot, another set of questions will be asked. These are needed to set the limits for the star display and to determine if you want point or continuous plots. Just like printing, you can plot for single or multiple days, with any number of days between plots. You can plot single points, (with the previous plot erased before the current one is plotted), or continuous plots, (where the points aren't erased but remain on the screen). After that you'll be asked for a scaling factor: 0 or 1-20. A scaling factor of zero will display the full star field, right ascension 0 to 24 hours, and declination 90 to −90 degrees. A scaling factor equal to or greater than 1 (a factor between zero and one is not allowed) displays another question, ''Enter center coordinates for R.A. and DEC.'' This will determine the center coordinates of the display, and is in hours and decimal hours, degrees and decimal degrees. The scaling factor you entered, along with the center coordinates, will determine the right and left, top and bottom limits of the display. The higher the scaling factor, the less of a constellation you'll see, but the greater the movement of the planet per plot. A scaling factor of 1 displays approximately 18 hours in right ascension and 180 degrees in declination, and a factor of 10 displays, approximately 2 hours in right ascension and 19 degrees in declination.

The only constellations in the star table are for the zodiac. If you want to increase the number of stars within the zodiac, or if you want to add more constellations, it's an easy process. The table is set up with four values per star. The first two are for right ascension in hours, minutes; the next two are for declination in degrees, minutes. The stars in the table don't have to be in any particular order. The whole table is read when the plot portion of the program is used. The only table requirements are the two values for right ascension and two values for declination. If the declination is negative, then both values for declination have to be negative. To end the table, four zeros are necessary — 0,0,0,0.

You may want to split this program to make one that just displays the stars on the screen. Just begin where the question for a scaling factor is asked, and delete everything else that isn't used. You can add more tables to the new program: one for galaxies, another for star clusters, another for nebulae, or even one for the Messier objects. The tables

you add will be whatever you need, and by adding more questions, you can display the different tables, either alone or combined.

Let's go through two examples of the program, first for figure 1, and second for figure 2. The first question that will be asked is if you want to display the same planets as your last run. Since this is the first run, enter N. Then it will ask "What planets do you want to display?" Enter a 1 for each planet. Then a starting date is asked. Use 11,1,1979. After that, it says "Enter the number of days to plot." Enter 150. Then it asks to print or plot. Enter a 1 to print. The screen will then clear, print the starting date and the plot day's value at the top of the screen, and then continue to print for the planets and the sun.

After finishing the page, it will pause and display "Press return for next display." After you press return it will start printing again, changing the plot day's value at the top of the page and the values for the planets and the sun. It will continue to do this until the plot day's value is equal to or greater than the day's that you wanted to print for. After that, it will ask you to press return to start again. When you press return, it will ask if you want to display the same planets as your last run.

For example 2, enter an N to the last question so that it will ask you which planets you want to display. Enter a 0 (zero) for all the planets except Mars. Enter 11,1,1979 for the starting date, 240 for the number of days to plot, and 10 for the number of days between plots. When it asks to print or plot, enter a 0 (zero) to plot. Three requests will then be made: the first, "enter 0 for point, or 1 for continuous plots." Since we want all the points to remain on the screen, enter 1 for continuous plots. The next question is the scaling factor. Enter a 5. After that will be the center coordinates. Since I already know that the planet Mars will be in the constellation Leo, enter 10.5 for right ascension, and 18 for declination.

When you do plots for other planets and you don't know where they will be, do the print program first and get the right ascension and declination from there. After entering the center coordinates, the screen will clear and a window will appear on the screen. After a few seconds the constellation Leo will appear as the star table is read, and any stars within the display limits will be plotted. A few more seconds will pass as the rest of the table is read. Once the end of the table is found, the program will beep to signal the start of the calculations.

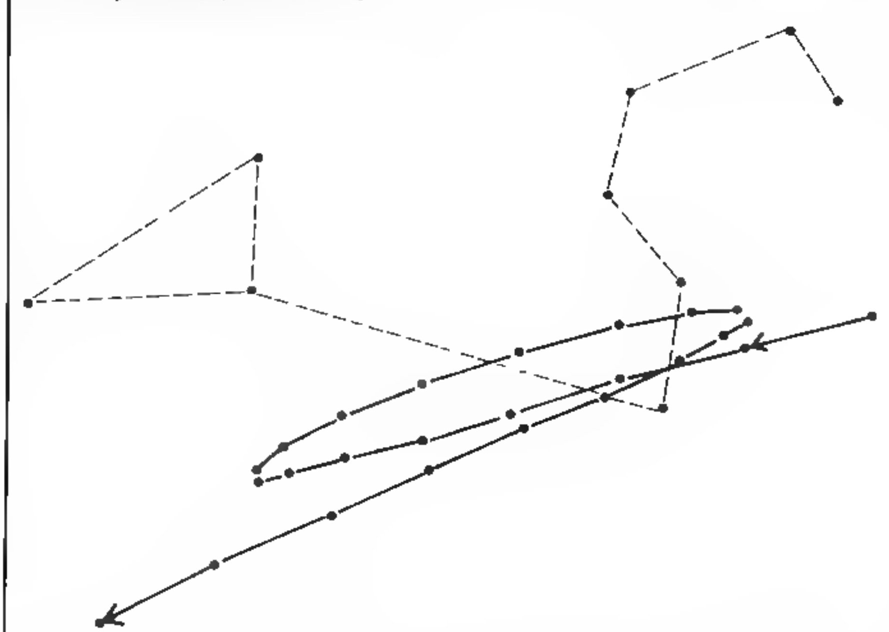
***Figure 1:* Example of the print routine for all planets, starting date 11/1/1979 for 240 days at 50-day intervals at the 150th day.**

Starting Date 11/1/1979 Plot Days 150

Earth	D-S.	92.8887	Sun	D-E.	92.8887
	A-S.	189.4489		R.A.	0 34.7
				DEC.	3 44.6
Mercury	D-S.	43.1581	Venus	D-S.	66.8181
	A-S.	245.1156		A-S.	140.7176
	D-E.	77.2616		D-E.	70.0302
	R.A.	22 55.3		R.A.	3 28.3
	DEC.	-8 7.1		DEC.	21 55
Mars	D-S.	154.4251	Jupiter	D-S.	502.2398
	A-S.	170.2956		A-S.	158.0192
	D-E.	73.2592		D-E.	425.652
	R.A.	9 56.5		R.A.	10 15.9
	DEC.	16 7		DEC.	12 9.5
Saturn	D-S.	875.6875			
	A-S.	174.1555			
	D-E.	785.842			
	R.A.	11 35.7			
	DEC.	5 15			

Press return for next display.

***Figure 2:* Example of the plot routine for Mars, starting date 11/1/1979 for 240 days at 10-day intervals, continuous plots.**



Since the planet Mars was the only planet picked, the program will calculate the positions of the earth and Mars. The position of the earth is always calculated, but only printed during the print option, (if you choose to print it). The program will continue to plot the position of Mars, beeping each time it starts a new sequence of calculations. It will plot 25 times — one for the starting date and 24 for 240 days, at 10-day intervals.

The program will then do a double beep to signal the end of the simulation and wait until you press return before starting a new sequence of questions. The purpose of the single beep at the beginning of the calculations is to indentify what planet is being plotted. The planets are plotted in their order from the sun. If you plot more than one planet in the same display, you can figure out which is which by the plotting order.

Since the date doesn't appear anywhere on the display for plotting, you can do a CNTL-C to stop the program, type "TEXT", and then return to see the starting date and the plot day's value. To continue, do POKEs to set graphics mode (– 16304) and display the secondary page (–16299), type "CONT" and return. The program will pick up where you left off. If you follow these examples, the results you get should match figure 1 at day 150 for printing, and figure 2 at the end of the plotting sequence. The solid and dotted lines in figure 2 were used to show the motion of Mars and the stars of the constellation Leo, and will not be in the actual display. Once you run the two examples to become familiar with the program, then you can enter any values for the questions to display whatever for whenever you want.

If you use the last program I wrote, you'll notice a difference. I don't have the assembler subroutines to do the plotting that I had done in the other one. Since writing the last program, I have gotten a disk drive and the disk version of floating point BASIC. If you have an Apple II Plus, this program should work as is. If you use the disk version of floating point BASIC you'll have to make minor changes.

Change the program loading address from $3000 to $6000, just beyond page 2 for Hi-Res graphics. This is done by getting into monitor, "reset" after bringing up your floating point BASIC. Change memory locations $67 and $68 to 01 60 (*67:01 60). Then change three locations at $6000 to zeros (*6000:00 00 00). *3D0G should get you back to floating point BASIC where you can enter the program. You can load this program at the normal location, ($3000), but the size of the program (approximately 9K) will put the end of it in the second page of Hi-Res graphics. Once you execute this program and hit the HGR2 command, the second page of Hi-Res graphics will clear, and so will whatever portion of the program is there. On the Apple II Plus, the end of the program will lie in page one of Hi-Res graphics, and since the program uses page two, no changes are needed.

For those Apple users who have the cassette version of floating point BASIC, all the necessary changes for using the assembler subroutines for plotting are in the article I wrote in the August '80 issue of MICRO. You'll have to set up a plotting subroutine and do a GOSUB where there is an HPLOT. Where there is an HCOLOR, you'll have to do a POKE 812,0 for a 0, or 255 for a 3.

Then for those users who don't have an Apple II or Hi-Res graphics, you can still use the print portion of this program. As I stated in the last article, if you have any problems or questions don't hesitate to call or write. (If you write please include an SASE to guarantee a response.)

Dave Partyka works as a programmer for The May Department Stores Co. Having worked for them for nine years, he finds programming the Apple II a relaxing diversion from programming the larger systems of the retail environment. He has had five articles published, this one being his third for MICRO. He uses the articles to buy accessories for his Apple II, offsetting the cost of expanding his system.

```
 10  REM   SOLAR SYSTEM SIMULATOR  # 2

 40  REM   DAVE PARTYKA
 45  REM   1707 N. NANTUCKETT DR.
 50  REM   LORAIN, OHIO   44053

100  GOTO 650
110  IF TY = 1 THEN 210
120  IF H > TP OR H < BT THEN 210
130  HCOLOR= 0
140  IF RG > LF THEN 180
150  IF F < RG OR F > LF THEN 210
160  HPLOT 279 - (F - RG) * SC,(TP - H) * SC
170  GOTO 210
180  IF F > LF AND F < RG THEN 210
190  IF F =  < LF THEN F = F + 360
200  HPLOT 279 - (F - RG) * SC,(TP - H) * SC
210  IF G > TP OR G < BT THEN  RETURN
220  HCOLOR= 3
230  IF RG > LF THEN 270
240  IF B < RG OR B > LF THEN  RETURN
250  HPLOT 279 - (B - RG) * SC,(TP - G) * SC
260  RETURN
270  IF B > LF AND B < RG THEN  RETURN
280  IF B =  < LF THEN B = B + 360
290  HPLOT 279 - (B - RG) * SC,(TP - G) * SC
300  RETURN
310 D = ZZ -  INT (ZZ / SRD) * SRD
320 B = Q - (D / SRD * Q2)
330  IF Y > 0 THEN RA = 270
340 RV = A - (P / (1 + E *  COS (B)))
350 V = PE / RV - EZ
360  IF V =  > 1 THEN V = VL
370  IF V =  <  - 1 THEN V =  - VL
380 VA =  -  ATN (V /  SQR ( - V * V + 1)) + T
390  IF D > SRD / 2 THEN VA = Q2 - VA
400 VA = VA + J
410 ZX = VA * T1 - C
420  IF ZX > 360 THEN ZX = ZX - 360
430  IF ZX < 0 THEN ZX = 360 + ZX
```

```
440 ZX = ZX / T1
450 LA =  SIN (ZX) * I
460 XA = RV *  COS (LA) *  COS (VA)
470 YA = RV *  COS (LA) *  SIN (VA)
480 ZA = RV *  SIN (LA)
490 XB = XA - X3:YB = YA - Y3:ZB = ZA - Z3
500 VA = VA * T1
510  IF VA > 360 THEN VA = VA - 360
520  IF EE = 0 THEN  RETURN
530 ED =  SQR (XB * XB + YB * YB)
540 X = XB
550 Y = YB *  COS (IN) - ZB *  SIN (IN)
560 Z = YB *  SIN (IN) + ZB *  COS (IN)
570 RA = 90
580  IF Y < 0 THEN RA = 270
590  IF X <  > 0 THEN RA =  ATN (Y / X) * T1
600  IF X < 0 THEN RA = RA + 180
610  IF X > 0 AND Y < 0 THEN RA = RA + 360
620 DZ = Z / ED
630 DC =  ATN (DZ /  SQR (1 - DZ * DZ)) * T1
640  RETURN
650 T = 1.5708:T1 = 57.2957795
660 IN = 23.434 / T1
670 Q = 3.14159265
680 Q2 = 6.2831853
690 VL = .99999999
700  HOME
710  PRINT "DO YOU WANT TO DISPLAY "
720  PRINT : PRINT "THE SAME PLANETS AS YOUR LAST RUN"
730  PRINT : INPUT "Y OR N ";A$
740  IF A$ = "N" THEN 790
750  IF A$ <  > "Y" THEN 710
760  IF S1 <  > 0 THEN 1590
770  IF SC <  > 0 THEN 2785
780  PRINT : PRINT "YOU HAV'NT PICKED THE PLANETS YET": PRINT :
     PRINT : GOTO 800
790  HOME
800  PRINT "CHOOSE THE PLANETS YOU WANT TO DISPLAY"
810  PRINT
820  PRINT "ENTER A 1 FOR YES, 0 FOR NO"
830  PRINT
840  REM  SPACIFIC VALUES FOR EACH PLANET
850  REM  S1=ORBITAL PERIOD: P1=A1*(1-E1*E1)/2
860  REM  E1=ECCENTRICITY: U1=P1/E1: K1=1/E1
870  REM  A1=MINIMUM + MAXIMUM DISTANCE FROM SUN
880  REM  J1=LONGITUDE OF PERIHELION IN RADIANS
890  REM  W1=DAYS FROM 0 DEGREES TO PERIHELION FOR 1980
892  REM  C1=ASCENDING NODE IN DEGREES
894  REM  I1=INCLINATION IN DEGREES / T1 TO CONVERT TO RADIANS
900  INPUT "DISPLAY MERCURY       ";ME
910 S1 = 87.969
920 E1 = .2056
930 A1 = 43.403 + 28.597
940 P1 = A1 * (1 - E1 * E1) / 2
950 K1 = 1 / E1
960 U1 = P1 / E1
970 J1 = 77.1 * Q / 180
980 W1 = 37.53
990 C1 = 48.1
1000 I1 = 7 / T1
1010  INPUT "DISPLAY VENUS         ";VE
1020 S2 = 224.701
1030 E2 = .0068
1040 A2 = 67.726 + 66.813
1050 P2 = A2 * (1 - E2 * E2) / 2
1060 K2 = 1 / E2
1070 U2 = P2 / E2
1080 J2 = 131.3 * Q / 180
1090 W2 = 140.5
1100 C2 = 76.5
```

(Continued on page 113)

```
1110 I2 = 3.4 / T1
1120  INPUT "DISPLAY EARTH         ";EA
1130 S3 = 365.256
1140 E3 = .0167
1150 A3 = 94.555 + 91.445
1160 P3 = A3 * (1 - E3 * E3) / 2
1170 K3 = 1 / E3
1180 U3 = P3 / E3
1190 J3 = 102.6 * Q / 180
1200 W3 =  - 3.82
1210 C3 = 0
1220 I3 = 0
1230  INPUT "DISPLAY MARS          ";MA
1240 S4 = 686.980
1250 E4 = .0934
1260 A4 = 154.936 + 128.471
1270 P4 = A4 * (1 - E4 * E4) / 2
1280 K4 = 1 / E4
1290 U4 = P4 / E4
1300 J4 = 335.7 * Q / 180
1310 W4 = 287
1320 C4 = 49.4
1330 I4 = 1.85 / T1
1340  INPUT "DISPLAY JUPITER       ";JU
1350 S5 = 4332.125
1360 E5 = .0478
1370 A5 = 507.046 + 460.595
1380 P5 = A5 * (1 - E5 * E5) / 2
1390 K5 = 1 / E5
1400 U5 = P5 / E5
1410 J5 = 13.6 * Q / 180
1420 W5 = 1608
1430 C5 = 100.24
1440 I5 = 1.3 / T1
1450  INPUT "DISPLAY SATURN        ";SA
1460 S6 = 10825.863
1470 E6 = .0555
1480 A6 = 937.541 + 838.425
1490 P6 = A6 * (1 - E6 * E6) / 2
1500 K6 = 1 / E6
1510 U6 = P6 / E6
1520 J6 = 95.5 * Q / 180
1530 W6 = 2090
1540 C6 = 113.51
1550 I6 = 2.49 / T1
1590  HOME
1600  PRINT "ENTER BEGINNING DATE? MM,DD,YYYY": INPUT "
        ";MM,DD,YY
1610 DF = (MM = 2) * 31 + (MM = 3) * 59 + (MM = 4) * 90 + (MM
= 5) * 120 + (MM = 6) * 151 + (MM = 7) * 181 + (MM = 8) * 212
+ (MM = 9) * 243 + (MM = 10) * 273 + (MM = 11) * 304 + (MM
= 12) * 334
1620 ZY =  INT (YY * 365 +  INT (YY / 4) + DD + DF + 1 -  INT
(YY / 100) +  INT (YY / 400) / 1)
1630  IF  INT (YY / 4) <  > YY / 4 THEN 1680
1640  IF  INT (YY / 400) = YY / 400 THEN 1660
1650  IF  INT (YY / 100) = YY / 100 THEN 1670
1660  IF MM > 2 THEN 1680
1670 ZY = ZY - 1
1680 ZY = ZY - 723180
1690 ZT =  - ZY
1700  PRINT : PRINT : INPUT "ENTER # OF DAYS TO PRINT/PLOT ";D
N
1710  PRINT : PRINT : PRINT
1720  INPUT "ENTER # OF DAYS BETWEEN PRINT/PLOTS ";DA
1730  IF DA <  > 0 THEN 1760
1740  PRINT : PRINT
1750  PRINT "0 NOT ALLOWED": GOTO 1710
1760  HOME
```


(Continued on next page)

(Continued on next page)

```
1770  INPUT "ENTER 1 TO PRINT, 0 TO PLOT  ";PL
1780  IF PL < > 0 AND PL < > 1 THEN 1760
1785  IF PL = 0 THEN  PRINT : PRINT "DO YOU WANT": PRINT : INP
UT "POINT (0) OR CONTINUOUS (1) PLOTS ";TY
1786  IF TY < > 0 AND TY < > 1 THEN 1785
1790  IF PL = 0 THEN  GOSUB 2750
1800  REM   EARTH
1810  HOME :EE = 0
1830 A = A3:P = P3:E = E3:PE = U3:EZ = K3:SRD = S3:J = J3:W =
W3:ZZ = ZY + W:C = C3:I = I3
1840  GOSUB 310:EE = 1
1845 X3 = XA:Y3 = YA:Z3 = ZA:R3 = RV:V3 = VA
1848  HOME
1850  VTAB 1: HTAB 1: PRINT "STARTING DATE ";MM;"/";DD;"/";YY;
"   PLOT DAYS ";ZT + ZY
1855  IF PL = 0 THEN  VTAB 23: PRINT "STARTING DATE ";MM;"/";D
D;"/";YY;"   PLOT DAYS ";ZT + ZY: PRINT "": GOTO 1980: REM  EM
PTY PRINT IS A CNTL-G (BELL)
1870  IF EA = 0 THEN 1980
1880  VTAB 2: HTAB 1: PRINT "EARTH D-S. "; INT (RV * 10000) /
10000
1890  VTAB 3: HTAB 7: PRINT "A-S. "; INT (V3 * 10000) / 10000
1900  REM   SUN
1910 XB =  - X3:YB =  - Y3:ZB =  - Z3:ED = R3
1920  GOSUB 540
1930  VTAB 2: HTAB 21: PRINT "SUN    D-E. "; INT (ED * 10000)
/ 10000
1940  VTAB 3: HTAB 28: PRINT "R.A. "; INT (RA / 15);"  "; INT
((RA -  INT (RA / 15) * 15) * 40) / 10
1950  IF DC < 0 THEN DC =  - DC:DB = 1
1960  VTAB 4: HTAB 28: PRINT "DEC. "; INT (DC);"  "; INT ((DC
-  INT (DC)) * 600) / 10
1970  IF DB = 1 THEN  VTAB 4: HTAB 32: PRINT "-":DB = 0
1980  REM   MERCURY
1990  IF ME = 0 THEN 2130
2000 A = A1:P = P1:E = E1:PE = U1:EZ = K1:SRD = S1:J = J1:W =
W1:ZZ = ZY + W:C = C1:I = I1
2010  GOSUB 310: IF PL = 1 THEN 2050
2020 F = F1:H = H1:B = RA:G = DC: GOSUB 110
2030 F1 = RA:H1 = DC: GOTO 2130
2040  IF PL = 0 THEN  GOSUB 110
2050  VTAB 6: HTAB 1: PRINT "MERC  D-S. "; INT (RV * 10000) /
10000
2060  VTAB 7: HTAB 7: PRINT "A-S. "; INT (VA * 10000) / 10000
2070  VTAB 8: HTAB 7: PRINT "D-E. "; INT (ED * 10000) / 10000
2080  VTAB 9: HTAB 7: PRINT "R.A. "; INT (RA / 15);"  "; INT (
(RA -  INT (RA / 15) * 15) * 40) / 10
2090  IF DC < 0 THEN DC =  - DC:DB = 1
2100  VTAB 10: HTAB 7: PRINT "DEC. "; INT (DC);"  "; INT ((DC
-  INT (DC)) * 600) / 10
2110  IF DB = 1 THEN  VTAB 10: HTAB 11: PRINT "-":DB = 0
2120  REM   VENUS
2130  IF VE = 0 THEN 2260
2140 A = A2:P = P2:E = E2:PE = U2:EZ = K2:SRD = S2:J = J2:W =
W2:ZZ = ZY + W:C = C2:1 = I2
2150  GOSUB 310: IF PL = 1 THEN 2180
2160 F = F2:H = H2:B = RA:G = DC: GOSUB 110
2170 F2 = RA:H2 = DC: GOTO 2260
2180  VTAB 6: HTAB 21: PRINT "VENUS  D-S. "; INT (RV * 10000)
/ 10000
2190  VTAB 7: HTAB 28: PRINT "A-S. "; INT (VA * 10000) / 10000
2200  VTAB 8: HTAB 28: PRINT "D-E. "; INT (ED * 10000) / 10000
2210  VTAB 9: HTAB 28: PRINT "R.A. "; INT (RA / 15);"  "; INT
((RA -  INT (RA / 15) * 15) * 40) / 10
2220  IF DC < 0 THEN DC =  - DC:DB = 1
2230  VTAB 10: HTAB 28: PRINT "DEC. "; INT (DC);"  "; INT ((DC
-  INT (DC)) * 600) / 10
2240  IF DB = 1 THEN  VTAB 10: HTAB 32: PRINT "-":DB = 0
2250  REM   MARS
```

(Continued on next page)

```
2260  IF MA = 0 THEN 2390
2270 A = A4:P = P4:E = E4:PE = U4:EZ = K4:SRD = S4:J = J4:W =
W4:ZZ = ZY + W:C = C4:I = I4
2280  GOSUB 310: IF PL = 1 THEN 2310
2290 F = F4:H = H4:B = RA:G = DC: GOSUB 110
2300 F4 = RA:H4 = DC: GOTO 2390
2310  VTAB 12: HTAB 1: PRINT "MARS  D-S. "; INT (RV * 10000) /
 10000
2320  VTAB 13: HTAB 7: PRINT "A-S. "; INT (VA * 10000) / 10000

2330  VTAB 14: HTAB 7: PRINT "D-E. "; INT (ED * 10000) / 10000
2340  VTAB 15: HTAB 7: PRINT "R.A. "; INT (RA / 15);"  "; INT
((RA -  INT (RA / 15) * 15) * 40) / 10
2350  IF DC < 0 THEN DC =  - DC:DB = 1
2360  VTAB 16: HTAB 7: PRINT "DEC. "; INT (DC);"  "; INT ((DC
-  INT (DC)) * 600) / 10
2370  IF DB = 1 THEN  VTAB 16: HTAB 11: PRINT "-":DB = 0
2380  REM   JUPITER
2390  IF JU = 0 THEN 2520
2400 A = A5:P = P5:E = E5:PE = U5:EZ = K5:SRD = S5:J = J5:W =
W5:ZZ = ZY + W:C = C5:I = I5
2410  GOSUB 310: IF PL = 1 THEN 2440
2420 F = F5:H = H5:B = RA:G = DC: GOSUB 110
2430 F5 = RA:H5 = DC: GOTO 2520
2440  VTAB 12: HTAB 21: PRINT "JUPTR  D-S. "; INT (RV * 10000)
 / 10000
2450  VTAB 13: HTAB 28: PRINT "A-S. "; INT (VA * 10000) / 1000
0
2460  VTAB 14: HTAB 28: PRINT "D-E. "; INT (ED * 10000) / 1000
0
2470  VTAB 15: HTAB 28: PRINT "R.A. "; INT (RA / 15);"  "; INT
 ((RA -  INT (RA / 15) * 15) * 40) / 10
2480  IF DC < 0 THEN DC =  - DC:DB = 1
2490  VTAB 16: HTAB 28: PRINT "DEC. "; INT (DC);"  "; INT ((DC
 -  INT (DC)) * 600) / 10
2500  IF DB = 1 THEN  VTAB 16: HTAB 32: PRINT "-":DB = 0
2510  REM   SATURN
2520  IF SA = 0 THEN 2640
2530 A = A6:P = P6:E = E6:PE = U6:EZ = K6:SRD = S6:J = J6:W =
W6:ZZ = ZY + W:C = C6:I = I6
2540  GOSUB 310: IF PL = 1 THEN 2570
2550 F = F6:H = H6:B = RA:G = DC: GOSUB 110
2560 F6 = RA:H6 = DC: GOTO 2640
2570  VTAB 18: HTAB 1: PRINT "SATN  D-S. "; INT (RV * 10000) /
 10000
2580  VTAB 19: HTAB 7: PRINT "A-S. "; INT (VA * 10000) / 10000
2590  VTAB 20: HTAB 7: PRINT "D-E. "; INT (ED * 10000) / 10000
2600  VTAB 21: HTAB 7: PRINT "R.A. "; INT (RA / 15);"  "; INT
((RA -  INT (RA / 15) * 15) * 40) / 10
2610  IF DC < 0 THEN DC =  - DC:DB = 1
2620  VTAB 22: HTAB 7: PRINT "DEC. "; INT (DC);"  "; INT ((DC
-  INT (DC)) * 600) / 10
2630  IF DB = 1 THEN  VTAB 22: HTAB 11: PRINT "-":DB = 0
2640 ZY = ZY + DA
2650  IF ZT + ZY > DN THEN 2700
2660  IF PL = 0 THEN 2690
2670  VTAB 23: HTAB 1: PRINT "PRESS RETURN FOR NEXT DISPLAY":
GET A$
2680  VTAB 23: HTAB 1: PRINT "                                 "
2690  GOTO 1830
2700 ZY = 0:DE = 0
2710 PRINT " "; " " : REM EMPTY PRINTS CNTL-G
2720  INPUT "PRESS ENTER TO START AGAIN";A$
2730  TEXT : RESTORE
2740  GOTO 650
2750  HCOLOR= 3
2760  PRINT : INPUT "ENTER FACTOR: 0 OR 1 - 20 ";SC
2770  IF SC <  > 0 THEN 2785
2780 RG = 0:LF = 360:BT =  - 90:TP = 110:SC = .75: GOTO 2890
2785  IF SC < 1 THEN 2760
```

```
2800  PRINT : PRINT "ENTER CENTER COORDINATES": PRINT
2810  PRINT " R.A.   DEC.": PRINT
2820  INPUT "HH.HH , DD.DD  ";R,D
2830 RG = R * 15 - 139 / SC
2840 LF = R * 15 + 139 / SC
2850 BT = D - 95 / SC
2860 TP = D + 95 / SC
2870  IF RG < 0 THEN RG = RG + 360
2880  IF LF > 360 THEN LF = LF - 360
2890  HGR2
2900  HPLOT 0,0 TO 279,0
2910  HPLOT  TO 279,191
2920  HPLOT  TO 0,191
2930  HPLOT  TO 0,0
2940  READ B,B1,G,G1
2950 B = B * 15 + B1 * .25:G = G + G1 / 60
2960  IF B = 0 AND G = 0 THEN  RETURN
2970  GOSUB 210: GOTO 2940
2980  REM  PISCES
2990  DATA  1,11,24,19,1,17,27,0,1,18,28,29,1,9,29,49,0,55,28,
43,0,47,27,26,0,53,26,56,1,28,15,5,1,43,8,54,1,59,2,31
3000  DATA  1,39,5,14,1,28,5,53,1,11,7,19,1,0,7,37,0,46,7,19,2
3,57,6,35,23,37,5,21,23,40,1,30,23,25,6,6,23,18,5,6,23,15,3,1,
23,24,0,59
3010  REM  ARIES
3020 DATA 1,51,19,3,1,52,20,34,2,1,25,42
3030  REM  PLEIADES
3040  DATA  3,42,24,8,3,42,23,57,3,42,24,18,3,43,24,13,3,43,24
,24,3,45,23,57,3,43,23,48
3050 REM TAURUS
3060  DATA  5,23,28,34,4,39,22,52,5,35,21,7,5,4,18,35,4,33,16,
25,4,26,15,51,4,17,15,31,4,23,17,49,4,26,19,4
3070  REM  GEMINI
3080 DATA  6,12,22,31,6,20,22,32,6,41,25,11,7,8,30,20,7,31,32,0,7,42,28,9,7,17,
22,5,7,1,20,39,6,35,16,27,6,42,12,57
3090  REM  CANCER
3100 DATA 8,14,9,20,8,18,24,11,8,30,20,37,8,29,18,16,8,42,18,20,8,40,21,39,8,
56,12,3,8,44,28,57
3110  REM  LEO
3120  DATA 9,43,24,0,9,50,26,15,10,14,23,40,10,17,20,6,10,5,17
,0,10,6,12,13,11,11,20,48,11,47,14,51,11,12,15,42
3130  REM  VIRGO
3140  DATA  11,43,6,49,11,48,2,3,12,17,-0,-23,12,39,-1,-11,12,
53,3,40,13,0,11,14
3200  DATA  13,7,-5,-16,13,23,-10,-54,14,13,-5,-46,14,40,-5,-2
7,14,44,2,6,13,59,1,47,13,32,-0,-20
3270  REM  LIBRA
3280  DATA  14,48,-15,-50,15,10,-19,-28,15,14,-9,-12,15,33,-14
,-37
3320  REM  SCORPIUS
3330 DATA 15,56,-25,-28,15,57,-22,-29,16,3,-19,-40,16,18,-25,-28,16,
28,-26,-19,16,33,-28,-7,16,47,-34,-12,16,48,-37,-58,16,50,-42,-17
50,-42,-17
3420 DATA  17,9,-43,-11,17,34,-42,-58,17,44,-40,-7,17,39,-39,-0,17,30,
-37,-4
3470  REM  SAGITTARIUS
3480  DATA  18,3,-30,-26,18,14,-36,-47,18,21,-34,-25,18,18,-29
,-51,18,25,-25,-27,18,43,-27,-3,18,52,-26,-22,18,59,-29,-57,19
,4,-27,-45
3570  REM  CAPRICORNUS
3580  DATA  20,15,-12,-40,20,24,-18,-23,20,36,-15,-8,21,3,-17,
-26,21,19,-17,-3,21,37,-16,-53,21,44,-16,-21
3650  DATA  21,40,-19,-6,21,34,-19,-41,21,26,-22,-2,21,24,-22,
-38,21,4,-25,-12,20,49,-27,-6,20,43,-25,-27
3720  REM  AQUARIUS
3740  DATA  22,3,-0,-34,22,23,1,7,22,26,-0,-17,22,33,-0,-23,22
,50.-7,-51,22,47,-13,-51,22,52,-16,-5,23,12,-6,-19,23,13,-9,-2
2,23,16,-9,-53,23,40,-14,-49
3830  REM  END OF TABLE (ZEROS)
3840  DATA 0,0,0,0
```

MICRO

MICRO Hardware Catalog

Name: **AME 1000 Microcomputer System**
Memory: 32K bytes of RAM, 22K bytes of ROM, 280K bytes of floppy storage
Language: BASIC with disk operating software, optional Assembler, EORTH, PI/65
Description: Tbc AME 1000 AIM 65-based, includes an enclosure, special motherboard, CRT controller (80×25) and Video 100 monitor, 2 single-sided double-density 5¼" floppy drives, controller and integrated disk operating software, Centronics-type printer controller and software driver, 32K bytes of dynamic RAM, 22K bytes of ROM-based operating software and a 5V-5A, ± 12V-1A power supply. Supports one additional Exorciser board and 6 RM 65 modules. Includes interface to 2 Braemar cassettes.
Price: $3,595 including AIM 65
Available: DYNATEM
20881 Paseo Olma
El Toro, CA 92630

Name: **Game Plus**
System: Apple II and II+
Hardware: Multi-Paddle/Joystick Adapter/multiplexer
Description: *Game Plus* is a hardware adapter which is plugged into the game I/O socket and allows the user to connect up to four sets of paddles or joysticks simultaneously. The unit features ultra-low current drain and is compatible with all existing software and hardware.
Price: $49.95
Available: Syntronics, Inc.
P.O. Box 601
St. Clair Shores, MI 48080

Name: **ColorMate**
System: SYM-1, KIM-1, AIM 65
Memory: 3K hytes 2114 static RAM
Description: A color video board based on the Motorola 6847 video display generator, the *ColorMate* offers nine modes of operation, ranging from alphanumeric to full graphic.
Price: $50.00 for PC board and manual (other options)
Available: MicroMate
P.O. Box 50111
Indianapolis, IN 46256

Name: **ZVM-121**
Description: Video monitor witb 12" green display. Styling and color compatible with Apple II and Apple III microcomputers. Display of 25 lines with 8×10 character matrix (640×250 pixels). Monitor uses standard NTSC video signal and connection to microcomputer is made via phono plug. Exterior controls include power, black level, contrast, horizontal and vertical adjustment, horizontal and vertical size.
Price: $160.00
Available: Zenith Data Systems dealers nationwide

Name: **I/O Selectric Interface Board**
System: Any system with serial or parallel I/O
Description: Input from serial or parallel I/O, output to I/O selectric. Board converts from ASCII and provides all timing.
Price: $35.00 ($15.00 for correspondence PROM)
Available: Computer Systems Consultants
1454 Latta Lane
Conyers, GA 30207·
(404) 483-1717/4570

Name: **High Density Static RAM/EPROM Module, GMS 6508**
Memory: Up to 24K bytes static RAM, sockets for up to 16K bytes EPROM/ROM
Description: Low cost, higb density memory module. Write protect, over voltage and reverse polarity connection, 1 MHz or 2 MHz operation. Compatible with Rockwell System 65, AIM 65, Motorola Exorciser. 24,576 bytes static RAM addressable in 8K-byte segments. Available in 8K, 16K, 24K versions. 6" × 9.75", +5 VDC power.
Price: $560.00, single piece quantity, 1 MHz
Available: General Micro Systems
1320 Cbaffcy Ct.
Ontario, Canada 91767

Name: **Model 2101**
Memory: Standard 2K buffer memory; 4K option
Description: Tbe *Model 2101* is a quiet, bi-directional, electrothermal printer that prints at 160 cps, so the user bas 120 cps throughput with 1200 baud communications. The model's 1×11 dot printhead generates 5×9 dot matrix characters with true upper/lower case and true underscore/overscore. A standard 80/132 selectability is great for multiple column formatting for financial and statistical applications. Only weighing 8.5 lbs, the printer can be used by OEM and end users alike.
Price: $1385
Available: Computer Devices Inc.
25 North Avenue
Burlington, MA 01803
1-800-225-1230

Name: **W7AAY ROM Board**
System: Synertek SYM-1
Description: Supplied completely assembled with instructions, this board plugs into the SYM's U23 socket. Two 24-pin ROMs or EPROMs can be plugged into tbe two sockets on the board. All addressing and selection jumpers are contained in the SYM's standard jumper area. Ideal for putting the two-chip versions of BAS or RAE into only one socket, or for adding two 2716 EPROMs.
Price: $16.00 each, ppd in USA
Available: Jobn M. Blalock
Blalock & Associates
P.O. Box 39356
Phoenix, AZ 85069

Name: **MI-J1 User Applications Connector**
System: Rockwell International AIM 65
Description: The MI-J1 applications connector is a compact printed circuit board that plugs directly into the J1 connector on tbe AIM 65 by a dual 22-pin gold plated edge connector. MI-J1 offers a numher of convenient and necessary connections to the AIM 65, and it also converts the 20 mA serial port to RS-232C signals. Connections to and from a cassette tape recorder are provided via audio plugs. Remote control lines are provided via solder pads. All of the user VIA signals (plus +5 and +24 VDC and GND) are provided via a 24-pin dip socket. Two serial ports (20 mA and RS-232C) are also offered via a 9-pin connector.
Price: $29.95 assembled and tested; $13.95 bare P.C. board
Available: Micro Interfaces, Inc.
P.O., Box 14520
Minneapolis, MN 55414

MICRO

MICRO Software Catalog

Name: **Write-On III**
System: Apple III
Memory: 96K
Language: Applesoft
Description: This word processor is designed to be easy to learn and use. *Write-On* has easy-to-remember commands; data can be inserted into the document as it is being printed, either from the keyboard or from *Write-On*'s data files; margin settings can be altered with a single keystroke. Up to 99 files can be merged at the time of printing to create large documents. The program can read, edit, and print text files created by other programs. In addition, a formatted "screen draft" can be examined on the video monitor before printing a document.
Price: $249.00
Author: Speicher Systems
Available: Rainbow Computing, Inc.
Mail Order Dept.
19517 Business Center Dr.
Northridge, CA 91324
(213) 349-0300

Name: **COMAL Starter Kit**
System: PET/CBM 8032 or 4032
Memory: 32K
Language: New
Hardware: Standard PET/CBM
Description: Complete 3-pass COMAL interpreter, user's manual, handbook, reference guide, diskette of sample programs, diskette of HELP files, and a one-year subscription to the *COMAL Companion* newsletter, complete with updates on COMAL. A complete package to get you started with COMAL.
Price: $47.95 includes COMAL interpreter, 2 disks, manual, handbook, guide.
Author: Borge Christensen and Len Lindsay
Available: COMAL Users Group
5501 Groveland Terrace
Madison, WI 53716

Name: **PFS Software Series**
System: Apple II
Memory: 48K
Language: Pascal/Assembly
Description: *PFS*, the first program in the series, is an easy-to-use program that solves information storage and retrieval problems. The user designs a form on the screen and then uses that form to enter, retrieve, modify, and print items of interest. *PFS: Report*, the second program in the series, is designed to work hand-in-band with *PFS* to produce tabular reports from existing PFS files. It sorts, calculates, totals, formats, and prints the information in your PFS files. And, like *PFS*, all of these features can be used without any programming.
Price: PFS - $95.00
PFS: Report - $95.00
includes program diskette, user's manual, and backup certificate
Author: John Page and Jean Seal
Available: Software Publishing Corporation
2021 Landings Drive
Mountain View, CA 94043

Name: **Aviculturist II Bird Classification Program**
System: Apple II, II+
Memory: 48K with DOS 3.3 or 3.2 and FP installed
Language: Applesoft BASIC
Hardware: Apple II or II+ computer, DOS 3.2 or 3.3 with controller card. Printer optional.
Description: Retrieve single or groups of birds from more than 1,000 species. 150 U.S. bird data included as sample file. Output selection menu gives 3 choices of known data (26 U.S. habitat zones, size in inches and # of colors). Program will retrieve name of bird, order, size in inches, status (protected, endangered, historical), and delete command. Up to 9 colors, diets, zone of habitat around U.S.A., nest sight, nest structure and number of eggs. Intended for junior high to high school students for biology and other ornithological studies. Any bird watchers can store findings, retrieve or compare with others.
Price: $35.00 for sample listing of 125 entries. $50.00 for complete data listing of 1000 U.S. birds.
Author: American Avicultural Art & Science Inc.
Available: American Avicultural Art & Science Inc.
3268 Watson Road
St. Louis, MO 63143

Name: **HSD STATS**
System: Apple II or II+, DOS 3.2
Memory: 48K
Language: Applesoft
Hardware: Optional — printer with serial or parallel interface, Silentype or graphics printer
Description: A menu driven statistics package which accepts 7 samples of 200 points each. The package offers descriptive statistics, 10 data transformations, frequency distribution, percentile ranks and points, 1 or 2 variable Chi-Square, Correlation Matrix for up to 7 variables, 3 t-tests file creation from single or combined samples, arithmetic inter-sample manipulations, printing of raw data and results, Hi-Res bargraph and scattergram. Easy to use.
Price: $99.95 includes disk, complete documentation, 3-ring binder
Author: Stephen Madigan, Ph.D.
Virginia Lawrence, Ph.D.
Available: Selected computer stores or
Human Systems Dynamics
9249 Reseda Boulevard
Suite 107
Northridge, CA 91324
(213) 993-8536

Name: **Grafpak.MX100**
System: Apple II
Memory: 32K minimum
Language: BASIC and 6502 ASM
Hardware: Disk II, Epson MX-100
Description: Dump either Hi-Res page horizontally at 1x or 2x, and vertically at 1x-4x. Dump both pages in perfect registration vertically at 1x-4x. Use normal or inverse inking, specify indent in inches. Compatible with most current I/O cards. Easy to use. Grafpaks available for Anadex, Integral Data, and other Epson printers.
Price: $44.95 for disk and guide
Author: Robert Rennard
Available: Your dealer or
SmartWare
2281 Cobble Stone Ct.
Dayton, Ohio 45431

Name: **Hayden Applesoft Compiler**
System: Apple II
Memory: 48K of RAM
Language: Applesoft
Hardware: Disk
Description: The *Hayden Applesoft Compiler* translates a standard Applesoft BASIC program into true machine code that runs from three to more than 12 times faster than normally interpreted code. The longer and more complex the original program, the greater

the increase in execution speed.
Price: $200.00 includes binder with complete documentation, disk and protection device
Autbor: Jonatban Eiten
Available: The Hayden Book Co.
50 Essex Street
Rochelle Park, NJ 07662

Name: **0-2. Option Strategy Tables**
System: PET
Memory: 8K
Language: BASIC
Hardware: PET/CBM
Description: Tables are printed giving prices of various option strategies of puts and calls, and tbeir combinations, for a list of underlying stock prices at three times to expiration.
Price: $15.00
Author: Claud E. Cleeton
Available: Claud E. Cleeton
122-109th Ave., S.E.
Bellevue, WA 98004

Name: **BASIC A+**
System: Atari 800/400
Memory: 32K, but 48K strongly recommended
Language: Extended BASIC for the Atari
Hardware: Atari 810 disk drive(s)
Description: *BASIC A+* maintains compatibility with Atari BASIC while adding simple but powerful access to the unique Atari system of Player/Missile Graphics. The serious programmer will appreciate capabilities of *BASIC A+* which are unmatched on most micros: business-oriented features such as PRINT USING, RECORD I/O, and BINARY I/O, structured programming through IF... ELSE... ENDIF, etc.; and much more.
Price: $80.00 BASIC A+; $150.00 BASIC A+, OS/A+, Assembler/Editor, and more
Autbor: Autbors of Atari BASIC
Available: Your local Atari dealer or Optimized Systems Software
10379-M Lansdale Ave.
Cupertino, CA 95014
(408) 446-3099

Name: **Integer BASIC Compiler**
System: Apple II or II+ with Integer or Language Card, DOS 3.3
Memory: 32K or 48K
Language: 6502 machine code
Hardware: Disk II (preferably two drives)
Description: Tbe *Integer BASIC Compiler* is a fully optimized compiler/run-time system for Apple's Integer BASIC. Numerous extensions include: CHR$, GET KEY functions, HOME, CLEAR, INVERT, NRML, FLASH, and 8 other new keywords for Hi-Res graphics. Supports a string length of 32767 (instead of 255). Compiler generates a mix of 6502 and 'GSL' code (more efficient than P-code). Speed/Space optimization selection — IBC is the fastest compiler/run-time system available for the Apple. Existing BASIC programs can be easily converted to run on any Apple II.
Price: $149.50 includes complete documentation. Software supplied on two disks.
Author: Christopher Galfo
Available: Galfo Systems
6252 Camino Verde
San Jose, CA 95119

Name: **Catalog**
System: PET/CBM
Memory: 8K and data storage
Language: BASIC
Hardware: No extra
Description: Keeps a catalog of books, records, stamps, etc. on tape and retrieves selectively by fields. Five fields provided, including one that holds up to 5 "category" identifiers for later retrieval.
Price: $16.95 includes cassette and manual
Available: Optimized Data Systems
P.O. Box 595
Placentia, CA 92670

Name: **Orbitron**
System: Apple II or II+
Memory: 48K
Language: Machine
Hardware: Disk drive, 13 or 16 sector controller
Description: This game places you in the center of an orbiting space station protected by a revolving force shield. The object is to fight off enemy forces which attempt to place killer satellites in orbit around your station. The game has seven levels of play, easy-to-use keyboard control, and super-fast high-resolution color graphics. The sound effects are incredible.
Price: $29.95 includes disk and documentation
Author: Eric Knopp - Presented by Sirius Software, Inc.
Available: Your local Apple dealer or software store

Name: **TransFORTH II**
System: Apple II, II+, and III
Memory: 48K/Apple II
Language: 6502 Machine Language
Hardware: Disk drive required
Description: TransFORTH II is an extended, fully compiled version of the FORTH language. It features floating point arithmetic; transcendental functions; strings and arrays; Hi-Res, Lo-Res, Turtlegraphics; and music.
Price: $125/introductory offer includes disk and manual
Author: Paul Lutus
Available: Insoft
259 Barnett Road, Unit #3
Medford, Oregon 97501

Name: **Compu-Read 3.0**
System: Apple II, II+; Atari 800
Memory: 48K
Language: Applesoft; Atari BASIC
Description: Contains a series of instructional modules which build learners' skills by strengthening the perceptual processes essential to competent reading.
Price: $29.95 each, includes documentation
Available: EDU-WARE Services
22222 Sherman Way, Suite 203
Canoga Park, CA 91303

Name: **SwordThrust**
System: Apple II or II+
Memory: 48K
Language: Applesoft
Hardware: One disk drive
Description: SwordThrust is an exciting new fantasy role-playing system designed for the Apple II. Unlike previous adventure series, SwordThrust is an integrated system of quests. The battles you fight and the gold you gather in one cave will affect you in the next. SwordThrust currently consists of the Master Diskette (which includes Adventure #1 and must be used to run any other adventure) and Adventures #2 through #4.
Price: $29.95 for the Master Diskette, $24.95 for the Adventure Diskettes
Author: Donald Brown
Available: CE Software
801 73rd Street
Des Moines, Iowa 50312

Name: **SEGS**
System: OSI
Memory: 24K
Software: OS65D
Hardware: Disk
Description: Adds segmentation commands to BASIC. Allows segment calls (like GOSUBs) to subroutines stored on disk. By nesting calls, large programs may be written and will run in 24K memory. Write to address below for more information.
Price: $25.00
Available: Universal Systems
2020 W. County Rd. B
Minneapolis, Minnesota 55113

MICRO

MICRO

Dr. William R. Dial
438 Roslyn Avenue
Akron, Ohio 44320

6502 Bibliography: Part XXXVII

1077. Recreational Computing 9, No. 5, Issue 50 (March/April, 1981)

Berggren, Stephen R., "Dozo," pg. 6-7.
An Apple Pascal program in Turtle graphics.

Conlan, Jim, "Programs, Problems and Solutions," pg. 29-31.
A collection of short routines for Atari micros.

1078. Apple-Dillo (February, 1981)

Teas, George, "Short Notes on Pascal," pg. 4.
Technique to get a PRINTER:list on your Pascal disk files.

Pickens, Rodney, "A Lagniappe," pg. 5.
Notes for Pascal users.

Teas, George, "Pascal Primer," pg. 6.
A library of special disks for Pascal users.

Bartley, David, "A Formatting Lister For Applesoft Programs," pg. 7-9.
A prettyprint routine for the Apple, with an accompanying example listing for a cryptogram program.

1079. KB Microcomputing 3, No. 5, Issue 51 (March, 1981)

Baker, Robert W., "Pet-Pourri," pg. 8-11.
Discussion of the Pro Kit #1 for the PET; Word Pro utility; etc.

Derfler, Frank J., "Dial-Up Directory," pg. 18-20.
More on the ABBS and PMS systems for bulletin boards based on the Apple.

Hubbard, Bill, "Cheap, Dumb Apple," pg. 65-66.
How to use an inexpensive modem board with the Apple.

Miles, Kenneth, "Auto-Menu for the Apple II," pg. 68-70.
Make the access to your programs on Apple Disks a little easier.

Antonelli, James A., "Faster Baud Rate for the Superboard II Cassette," pg. 112-114.
A simple hardware mod to allow you to save and load at 600 baud.

Ratliff, Gary L., "PET Shorthand Complete," pg. 144-148.
Abbreviated keywords for early PETs.

1080. The Target (November/December, 1980)

Sellers, George, "Music," pg. 2-3.
A translation for the AIM from a KIM routine, showing how to use the 6522 timer in place of the KIM's 6530 timer. With two song tables.

Kaynor, Doug, "Tape Recorder," pg. 3.
A hardware mod for the AIM tape system.

Hall, Dale L., "Memory Test," pg. 9.
A dynamic memory test in zero page for the Aim-65.

1081. The Apple Barrel 4, No. 1 (January, 1981)

Gilbreth, Lee, "File Cabinet Valid Feature," pg. 4-6.
Some further additions for this useful utility.

1082. L.A.U.G.H.S. 3, No. 2 (February, 1981)

Roe, David, "There is a Better Way," pg. 3-7.
All about saving a Hi-Res picture in Pascal on the Apple.

Finn, Mike, "Outputting Strings in Assembly Language Programs," pg. 8-11.
An instructional article on machine language.

1083. Interface Age 6, Issue 3 (March, 1981)

Baker, Al, "Game Corner," pg. 22-26.
Music program for the Atari microcomputer.

Moore, Herb, "Sounds of the Atari... In BASIC," pg. 56-60.
Fundamentals of programming the Atari for music generation.

1084. Creative Computing 7, No. 3 (March, 1981)

Greenberg, Gary, "Wordpro 1 vs. CMC," pg. 26-27.
Two programs for the PET on word processing.

Ripin, Peter and Narayan, Krishna, "A PET Lizard," pg. 78.
A hand-eye coordination game for the PET.

1085. The Apple Orchard 1, No. 3 (Winter, 1980-1981)

Golding, Val J., "Print FRE(ed)," pg. 7.
Comments on copying copyrighted software.

Rowe, Pete, "A BASIC to Machine Language Interface," pg. 11-19.
Joining an interactive BASIC language shell to a machine language, computational viscera.

Clardy, Robert C., "Applesoft Program Listing Formatter," pg. 21-27.
Tidy up your program listings and make them easier to read.

Hyde, Randy, "Locations of Interest to Pascal and 6502 Users," pg. 28.
The listed addresses should be of interest to Apple/Pascal users.

Haines, Ron, "Some Notes About the UCSD Assembler," pg. 35, 39.
Notes for Apple/Pascal users.

Eisenberg, J.D. and Hertzfeld, A.J., "Inside the Silentype Firmware," pg. 43-52.
Useful information for Silentype printer users, interface routines, etc.

Wozniak, Steve, "Binary-To-Decimal Shortcut," pg. 52-53.
A better and faster conversion routine for the Apple.

Crossley, John, "Tabbing with Apple Peripherals," pg. 55.
Tabbing routines for Apple printers.

Hoyt, Jim, "Apple Writer and the Telephone," pg. 56-57.
A quick and easy way for transmitting Apple Writer files over the telephone to another Apple.

Kellner, Jo and Kellner, Charles, "Converting Strings to Numeric Variables," pg. 59-61.
A unit for the Apple/Pascal system.

Kellner, Jo, "Pascal Run-Time Errors," pg. 62.
A simple program for Apple/Pascal users.

(Continued on next page)

Brown, Tom, "POKE Salad," pg. 69-71.
A formatter routine for the Apple.

Evans, Frank, "Using USR," pg. 75.
An instructional article on the USR function with an example program.

1086. Peek(65) 2, No. 2 (February, 1981)

Holt, Ken, "Assembler Programmer's Guide to OSI Board Interfacing," pg. 9-12.
Information for those designing I/O interfaces for the OSI systems.

Jones, David A., "Cassette Corner," pg. 15.
Notes on CHECKBYTE; viewing contents of another tape about to be loaded; terminal operations; etc.

1087. The G.R.A.P.E. Vine 2, No. 1 (February, 1981)

Lawson, Steve, "Direct Disk Read and Write," pg. 4-5.
Routines to directly write to and read back from specified disk sectors, including an example program "Verse by Verse," for the Apple.

Trusty, Doug, "GHR2 Fonts," pg. 6.
A program to convert a 'Higher Text' large-size font to a 'GHR2' large-size font, for the Apple.

1088. The Apple Peel 3, No. 3 (March, 1981)

Brown, Tom, "POKE Salad," pg. 6.
Telephone dialing routine; sorting phone lists; etc.

Donahue, Thomas D., "Printer Chatter," pg. 9.
How to interface the Apple II to a Radio Shack Daisy Wheel II printer using the Centronics Apple Parallel Board.

1089. Nibble 2, No. 1 (February, 1981)

Hill, Alan G., "Amper-Reader," pg. 23-37.
A machine language routine for ROM Applesoft that will improve the read time for string array variables. An instructional article and program, demo, etc.

Riley, Kevin D., "Apple Communicator for the Handicapped," pg. 47-51.
Program listing and hardware suggestion for allowing the handicapped to communicate via the Apple.

Reynolds, William III, "Tricks While Using Text Files with BASIC Programs," pg. 53.
Tips on using the text files.

Wagner, Roger, "Use of HGR vs. HGR2 on the Apple II," pg. 55.
How to decide which HGR page should be used on the Apple.

Harrell, Keith, "Pascal Pointers and Principles," pg. 63-69.
Procedures and functions and how they can be used in writing and debugging Pascal programs.

Szetela, David, "Auto Run and Tape Software Protection," pg. 73.
Information on techniques used to protect software.

Harrell, Keith, "An 80-Column Display for the Apple: A Reality or a Fantasy?", pg. 77-78.
A comparison of available 80-column boards for the Apple.

1090. Byte 6, No. 3 (March, 1981)

Zimmermann, Mark, "A Beginner's Guide to Spectral Analysis, Part 2," pg. 166-198.
A 6502 assembly language program for the PET in which any reasonable function may be expressed as a sum of sines and cosines of various frequencies.

1091. Apple/Sass (March, 1981)

Fan, Kenny, "Printing the Unprintables Within Print Statements," pg. 7-8.
Three approaches to this task on the Apple.

Norris, Paul, "Using Your Printer from Pascal," pg. 16-17.
A 'how-to' article for Apple/Pascal users.

Bromback, Randy, "Once Upon A Function," pg. 19-20.
An instructional article on the use of the Applesoft user-defined function, with examples.

Murdoch, David M., "Ampersand Catalog," pg. 21.
A short routine to catalog with just the ampersand key, on the Apple.

Hartley, T., "Extended Catalog," pg. 25.
A short program to continue the catalog on Apple Disks so up to 181 files can be entered on the disk.

1092. Dr. Dobb's Journal 6, Issue 3, No. 53 (March, 1981)

Whitlock, Howard W., "Analysis of the Use of the 6502's Opcodes," pg. 11-13.
Frequency of use of the 151 different opcodes of the 6502 microprocessor, based on the 6,355 or so instructions in PET BASIC.

1093. Microcomputer Index 1, No. 4 (October-December, 1980)

A subject index covering over 1,000 microcomputer magazine articles, many relating to the 6502 microprocessor.

1094. OKC Apple Times @ Issue 2 (February, 1981)

Simpson, David A., "Meet Special Characters."
Four ways to print those special characters (underline, backslash, left bracket) on your Apple.

1095. Call -A.P.P.L.E. 4, No. 2 (February, 1981)

Pump, Mark, "DOS Internals: An Overview," pg. 8-12.
The first part in a continuing major series on the Apple DOS.

Thompson, J.C., "Universal Update," pg. 14-16.
A routine that updates any 13-sector diskette so it will boot in either a 13- or 16-sector environment.

Balderson, Robert, "Converting Dakin5's Patcher to DOS 3.3."
A patch for the Dakins Patcher routine to enable it to read all 16 sectors of any DOS 3.3 diskette on the Apple.

Landsman, Richard and Horton, Richard, "DOS 3.3/3.2 Boot Switch," pg. 22.
A hardware mod that allows the CTRL key to be used to switch between Apple diskette controller proms.

Crawford, Tom, "Applesoft Random Function," pg. 24-27.
An article dealing with a characteristic of the Applesoft Random function in which a malfunction was found after about 10,000 accesses.

DeGroat, Ron, "Seeing Double with Pascal Graphics," pg. 30-33.

MICRO

MICRO

Advertiser's Index

Next Month in MICRO

Games Bonus Section

- **Saucer Launch** — A game exploring the special hardware of the Atari 800. You as the gunner's apprentice and one of the few remaining survivors of the Starfleet patrol must destroy at least 60 percent of the attacking squadron of robot saucers while they are being launched.
- **Lunar Lander Animated Graphics in BASIC for the Color Computer** — This article on the TRS-80 Color Computer uses the game Lunar Lander to demonstrate high-speed animation through the use of Microsoft's Extended Color BASIC.
- **The Ultimate Ping-Pong for the PET** — This machine language version of the popular ping-pong game, playable on any PET, serves as an example of high-speed animation techniques.
- **Othello** — This game of strategy for two players using an Apple II was designed to simulate the popular board game of the same name.

Apple Bonus Section

- **Apple Bits: Part 3** discusses how to create "animations" for the low resolution screen.
- **ASCII Dump for the Apple** presents an assembly language program that extends the "Examine Memory" routine in the Apple monitor.
- **FDGEN** describes a program for building Applesoft subroutines to handle keyboard input, display output, file input/output/update, sorts and PRINT USING.
- **ROM Applesoft PRINT USING** helps readers easily format the output of numeric variables with decimals lined up, trailing zeros added, and commas inserted.

Other November Articles

Other articles scheduled for November include *OS-9 and the 6809: Revolutionary Tools; General BASIC-Machine Language Interface* (for the AIM); and *Pascal Tutorial*, the first in a three-part series designed with the beginner in mind.